

NASA-CR-177390
19860022352

A Reproduced Copy
OF

NASACR-177,390

Reproduced for NASA
by the
NASA Scientific and Technical Information Facility

LIBRARY COPY

FEB 11 1967

LANGLEY RESEARCH CENTER
LIBRARY, NASA
HAMPTON, VIRGINIA



1. Report No. NASA CR 177390		2. Government Accession No.		3. Recipient's Catalog No.	
4. Title and Subtitle A 3-Component Laser-Doppler Velocimeter Data Acquisition and Reduction System				5. Report Date January 1986	
				6. Performing Organization Code	
7. Author(s) L. C. Rodman, J. H. Bell and R. D. Mehta				8. Performing Organization Report No.	
9. Performing Organization Name and Address Stanford University Stanford, CA 94305				10. Work Unit No.	
				11. Contract or Grant No. NCC-2-294	
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington D. C. 20546				13. Type of Report and Period Covered Contractor Report	
				14. Sponsoring Agency Code RTOP No. 505-60-31	
15. Supplementary Notes Point of Contact: Technical Monitor, L. S. King, MS 227-8 Ames Research Center, Moffett Field, CA 94035 (415) 694-6616, FTS 464-6116					
16. Abstract This report describes a Laser Doppler Velocimeter capable of measuring all three components of velocity simultaneously in low-speed flows. All the mean velocities, Reynolds stresses, and higher-order products can then be evaluated. The approach followed is to split one of the colors used in a 2-D system, thus creating a third set of beams which is then focused in the flow from an off-axis direction. The third velocity component is computed from the known geometry of the system. In this report, the laser optical hardware and the data acquisition electronics are described in detail. In addition, full operating procedures and listings of the software (written in BASIC and ASSEMBLY languages) are also included. Some typical measurements obtained with this system in a vortex/mixing layer interaction are presented and compared directly to those obtained with a cross-wire system.					
17. Key Words (Suggested by Author(s)) laser velocimetry, wind tunnel measurements				18. Distribution Statement Unclassified-Unlimited STAR Category 34	
19. Security Classif (of this report) Unclassified		20. Security Classif (of this page) Unclassified		21. No. of Pages	
				22. Price*	

NASA CONTRACTOR REPORT 177390

Ames

JAN 7

11-27050

A 3-Component Laser-Doppler Velocimeter Data Acquisition and Reduction System

L. C. Rodman
J. K. Bell
R. D. Mehta

(NAS-CP-177390) A 3-COMPONENT
LASER-DOPPLER VELOCIMETER DATA ACQUISITION
AND REDUCTION SYSTEM (Stanford Univ.) 82 p
CSCL 70D

N86-31824

Unclas
G3/34 43099

CONTRACT NCC-2-294
January 1986

NASA

N86-31824#

NASA CONTRACTOR REPORT 177390

A 3-Component Laser-Doppler Velocimeter Data Acquisition and Reduction System

L. C. Rodman
J. H. Bell
R. D. Mehta
Stanford University
Stanford, California

Prepared for
Ames Research Center
under Cooperative Agreement WCC-2-294
January 1986



National Aeronautics and
Space Administration

Ames Research Center
Moffett Field California 94035

TABLE OF CONTENTS

	page
ABSTRACT	iii
LIST OF FIGURES	iv
NOMENCLATURE	v
1. INTRODUCTION	1
2. OPTICAL SYSTEM AND SIGNAL PROCESSING HARDWARE	3
2.1 Optics Table	4
2.2 Transmitting Optics	5
2.3 Receiving Optics	6
2.4 Signal Processing Hardware.....	6
2.5 Computer Interface	7
2.6 HP 9845A Desk-Top Computer	8
3. DATA ACQUISITION AND REDUCTION SOFTWARE	8
3.1 Data Acquisition Software	9
3.2 Data Reduction Software	10
4. OPERATING PROCEDURES	15
4.1 Alignment Procedures	15
4.2 Signal Processing	17
5. INHERENT PROBLEMS	19
6. SAMPLE RESULTS	21
7. CONCLUDING REMARKS	22
APPENDIX -- SOFTWARE FOR THE HP 9845B COMPUTER	23
ACKNOWLEDGEMENTS	67
REFERENCES	68
TABLE 1	69
FIGURES	70

This Page Intentionally Left Blank

LIST OF FIGURES

- Fig. 1 Schematic of the optics table layout.
- Fig. 2 Schematic of the transmitting optics.
- Fig. 3 Details of the probe volumes in the 3-D system.
- Fig. 4 Schematic of the receiving optics.
- Fig. 5 Evaluation of the effective probe length.
- Fig. 6 Schematic of the signal processing electronics.
- Fig. 7 Computer interface simplified block diagram.
- Fig. 8 Block diagram for data acquisition program.
- Fig. 9 Block diagram for data reduction program.
- Fig. 10 NASA LDV-A/D computer interface connections and settings.
- Fig. 11 Experimental rig.
(a) Overall schematic
(b) Details of boundary layer trips and coordinate system
- Fig. 12 Secondary velocity plots.
(a) Cross-wire measurements
(b) LDV measurements
- Fig. 13 Comparison of X-wire and LDV measurements in a vortex/mixing layer interaction.

1. INTRODUCTION

Our ability to understand and model turbulent flows still relies heavily on the availability of accurate measurements of mean and fluctuating quantities within the flow. Until recently, the hot wire was the only reliable tool available for the measurement of fluctuating velocities. In fact, almost all of our present knowledge about turbulent flows is based on measurements made with hot wires. In relatively simple flows (moderately two-dimensional with small cross-flows), reliable and accurate hot wire measurements are now possible with fully automated data acquisition and reduction systems which minimize errors due to drifts in calibrations. An example of such a system is given in Ref. 2.

However, as we turn our attention towards more complex turbulent flows, a need for more sophisticated measurement techniques has become apparent. These complex flows include those with compressibility effects, strong three-dimensionality (with steep mean gradients), flow reversals, and time-dependent behavior. Since about the mid-sixties, the most popular alternative tool for measuring mean and fluctuating velocities in turbulent flows has been the Laser Doppler Velocimeter (LDV).

The most popular LDV arrangement used for wind tunnel measurements is the dual beam or fringe method. In this method, one of the laser lines is split into two lines of equal intensity which are then focused through a lens so that they cross over at the focal point. The flow is seeded with small particles (typically less than 3-4 μm in diameter) which follow the fluid motion. As these particles pass through interferometric fringes created by the crossed laser beams, light is scattered off them which is received by a photodetector. The frequency of this scattered light, along with a knowledge of the fringe pattern formed by the laser beams, provide the means to calculate the velocity of the particle. The fringe method, especially in

the forward scatter, off-axis mode, generally offers the best signal-to-noise ratios and spatial resolution.

Although LDV systems are somewhat complex and tiresome to set up, they have certain advantages over hot wires for turbulence measurements. The fact that Laser Doppler Velocimetry is non-intrusive is especially beneficial in the measurement of unstable flow phenomena which are very sensitive to the presence of measurement probes. In certain situations, LDV systems can also provide greater spatial resolution and better directional discrimination than hot wires. This makes it possible to use LDV systems for the measurement of separated flows. Since an LDV measures the velocity directly, independent of the thermodynamic properties of the flow, it is particularly attractive for velocity measurements in compressible flows. Furthermore, the calibration converting the frequency to velocity is linear and easy to implement in software. This feature also allows for uniform sensitivity in measuring both moderate and high turbulence intensities.

Two-color LDV systems capable of measuring two components of velocity simultaneously are now being widely used. However, the main interest in the present investigation was to study three-dimensional interactions where it is desirable to obtain measurements of all three velocities. Hence, the first objective was to develop a laser velocimeter system capable of measuring all three components of velocity simultaneously so that all six components of Reynolds stress may be computed. Another objective was to compare these LDV measurements directly with those obtained with hot wires in flow fields where both techniques are expected to perform satisfactorily. The purpose of this is to evaluate the performance of the system quantitatively and objectively.

The approach followed in the present investigation was to convert an existing two-component LDV system into one capable of measuring all three components of velocity simultaneously. The system utilizes two wavelengths (488.0 and 514.5 nm) from a 4-watt Argon-Ion laser. The main four-beam matrix measures

u and v directly. The green line in the four-beam matrix is split (in half) using a dichroic filter and directed over the top of the traverse mechanism with mirrors, giving the third beam pair for the measurement of the w-component. This pair of beams (with rotated polarization) measures $w \sin 45^\circ + v \cos 45^\circ$. Scattered light is collected in the off-axis forward scatter mode using two collection lenses.

Signal processing is accomplished with single-particle burst counters, and the validated data are multiplexed through a "home-built" interface to an HP 9845B desk-top computer. Some selected first and second order products are reduced on-line, and the raw data is dumped onto floppy disk. An off-line program reduces the data, giving up to third order quantities and also plots histograms of the raw data for each channel. The software includes the capability to filter out noise by examining the histograms.

The optical and signal processing hardware is described in Section 2. The data acquisition and reduction software is described in Section 3 and detailed operating procedures are given in Section 4. Some problems, inherent to 3-component LDV systems, are presented in Section 5. Sample results from an experiment measuring mean and turbulence quantities in a vortex/mixing layer interaction are compared directly to results obtained using crossed hot-wire anemometry in Section 6, and concluding remarks are presented in the final section. Complete software listings written in BASIC and ASSEMBLY languages to run on the HP 9845B desk-top computer are included in the appendix.

2. OPTICAL SYSTEM AND SIGNAL PROCESSING HARDWARE

The hardware for the 3-component LDV system can be divided into three categories: the optical system, the signal processing instrumentation, and the computer. The LDV optics consist of the optics table, where the laser beam is split into green and blue beam pairs, the transmitting optics, where the beams are

directed into the flow field, and the receiving optics, where the scattered light is picked up by photodetectors. The signal processing instrumentation consists of amplifiers, filters, burst counters and a computer interface. . . sampling procedures are all computer controlled.

2.1 Optics Table

The optics table consists of the laser and all the optical elements needed to provide the necessary four-beam matrix. Fig. 1 shows a schematic of the arrangement with component numbers as referred to in this section. A 4-watt Argon-Ion laser (Lexel Model 95) is used to produce the main beam. The beam then passes through a collimator (1), which ensures that the beam waist occurs at the focal point of the transmitting lens.

The collimated beam is passed through a color separator box, which consists of a polarization rotator (2), an attenuator (3), and a pair of high dispersion Brewster angle prisms (4) which are used to separate the multi-line beam into two colors, blue (488 nm) and green (514.5 nm). These two beams are then reflected across the box by mirror (5), and out of the box by mirrors (6, 7).

Following the color separation, the green beam's polarity is rotated to horizontal (8), and the beam is split into two beams in the vertical plane (9). Most beam splitters prefer this type of perpendicular polarization for maximum efficiency. Using the beam displacer (10), the blue beam is then moved to the center of the optics, and its polarity is rotated to vertical (11). The blue beams are split in the horizontal plane (12). At this point, the four beams are each displaced 25 mm from the optical axis.

Two Bragg cells (13, 14) are used to shift the frequency of one beam from each pair. The unshifted beam passes through an optical rod so that the path lengths are matched. The frequency is shifted by a fixed amount of 40 Mhz. This shift creates

a moving system of fringes at the beam intersection point, allowing for directional discrimination of the velocity. Frequency shifting also helps to reduce the percentage of frequency change in highly turbulent flow, to reduce fringe bias, and to optimize frequency, thus enabling easy removal of the pedestal by high pass filtering.

The four beams are then passed through a beam steering module (15). The module consists of a set of wedge prisms that can be independently rotated about the beam axis to steer the shifted beam in any direction. This allows a more precise alignment of the beams. The beams are finally passed through a beam displacer (16) to reduce the beam spacing to 13 mm and a rotating prism (17) before leaving the optics table. The rotating prism enables the four-beam matrix to be rotated independently so that the beams may be aligned relative to the tunnel axes.

2.2 Transmitting Optics

The transmitting optics (Fig. 2) are mounted on a traversing mechanism with three degrees of freedom. The traverses are driven by individual stepper motors. The four beams from the optics table are directed by a set of five mirrors through a dichroic filter before being focused by a 380 mm (15 inch) focal length lens. This main four-beam matrix measures u and v directly. The dichroic filter, set at an angle of about 30 degrees to the incoming beams, splits the green beams in half, which provides the third beam pair for the third velocity component, w . This third beam pair is directed over the top by mirrors, passed through a polarization rotator (giving it a different polarity than the main-axis green beam pair) and then focused at the focal point of the main beam set. Since this third beam pair intersects the main measuring volume at a 45° angle to the main axis, it measures v and w with equal sensitivity, with the measured component being $w_v = (v \cos 45^\circ + w \sin 45^\circ)$. Since v is measured directly, w can be evaluated using the equation:

$$w = \frac{w_v - v \cos 45^\circ}{\sin 45^\circ} \quad (1)$$

Typical probe volume dimensions for each beam pair in the present configuration are 10 mm in length and 0.2 mm in diameter (Fig. 3). However, the actual "viewed" dimensions are reduced considerably, as discussed below, in Section 2.3.

2.3 Receiving Optics

The detector system (Fig. 4) is in the off-axis forward scatter mode. The receiving optics are mounted on a traversing gear, also run by stepper motors, which moves synchronously with the transmitting optics. Scattered light is collected by two 380 mm (15 inch) focal length lenses. The collimated light is passed through filters to separate the colors and the off-axis line is additionally passed through a polarization filter to avoid collecting light scattered by the main axis green pair. The collected light is then focused by 250 mm (10 inch) focal length lenses onto pin-hole apertures mounted in front of the three photomultiplier tubes. The collection angle and diameter can be adjusted to select the effective (viewed) probe length (Fig. 5). In the present set-up, a collection angle of 30 degrees and an aperture diameter of 0.5 mm were used to give an effective length of about 1.5 mm.

2.4 Signal Processing Hardware

The signals from the photomultiplier tubes are amplified and relayed to the signal processors via high-pass filters and mixers (Fig. 6). The amplifiers used are EIN model 403LA with a fixed gain of 37 dB and the filters are Allen Avionics F2440 with a fixed high-pass cutoff of 10 MHz. The filtered signals are mixed electronically with sine waves from three Tektronix SG503 Levelled Sine Wave Generators. The mixer is commercially available from Hewlett-Packard, model 10534A. The mixing procedure

is necessary for low-speed flows, where the actual Doppler frequencies are small compared to the Bragg frequency of 40 MHz. So in order to reduce the effective measured frequency and hence improve the counter resolution, the incoming signals are mixed with sine waves of known frequency.

The mixed signal is fed into single particle burst counters (Macrodyne model 2096-2 and 3003) via high-pass/low-pass filters and an amplifier (x10). The counters measure the zero crossings of the Doppler signal, which is related to the Doppler frequency by the range set on the Macrodyne. The range is set manually based on expected flow velocities, since it limits the frequencies that the processor can see for the given 10 bits of resolution.

The processors use two checks to validate a Doppler signal. The first check is the usual 5/8 comparison, where the processor checks the frequency for 5 zero crossings against that for 8 crossings. The second check is the multi-sequence check. Positive and negative thresholds are set on the signal, and a validated output is permitted only if, for all eight fringe crossings, the signal passes through a positive threshold, a zero level, and a negative threshold in the proper sequence. The digital data (consisting of a 10 bit data word with 3 bits giving the range) and a sync pulse (produced every time the front end of a valid burst is detected) are passed to the computer interface.

2.5 Computer Interface

A NASA LDV-A/D computer interface (CI) is used to transfer data from the LDV signal processor to the computer (Fig. 7). The CI can interface either digital or mixed analog and digital data to an HP 9845B desk-top computer. The CI consists of an eight-channel multiplexer, a four-channel A/D converter, and an event synchronizer with time interval counter.

For use with the LDV, the inputs to the CI are all digital. Six of the eight words come from the processor, and two are time and status words from the synchronizer. The inputs are

multiplexed to a single digital data channel output.

The CI can accept data from the three processors in either random mode or sync mode. In random mode, the CI will accept data inputs when an event occurs on any of the three channels. A dead time (between 5 and 50 μ s) is set in this mode, which controls the minimum time between samples to ensure that a given particle is sampled only once. In sync mode, the CI will accept data inputs when all three processors sample simultaneous events. In this mode, a coincidence time (between 5 and 50 μ s) has to be set, which determines the time window within which all three events must occur. A detailed description of the CI can be found in Ref. 5.

2.6 HP 9845B Desk-top Computer

Multiplexed data are passed to the HP computer from the NASA LDV-A/D using the HP 98032A high-speed 16-bit parallel interface. Jumpers labeled "9,B,D" are connected inside the 98032A for proper operation with the LDV-A/D. A select code of 10 (screw setting on the 98032A) is set for use with the software described in the appendix. A data buffer of 24 kbytes is provided in the memory of the HP 9845B desk-top computer for storage of up to 2000 samples obtained from one measurement location. Since three of the six data words passed for each sample are merely monitored and discarded as described above, 12 kbytes of raw data remain to be stored for each point. An HP 9895A floppy-disk drive is used for archival storage of raw data. The buffered data are written in real time to a sequential-access floppy-disk file. Enough header information is written to each file to identify the run, and to reproduce calibration tables.

3. DATA ACQUISITION AND REDUCTION SOFTWARE

Data acquisition and reduction on the HP 9845B is done via

two programs. The data acquisition program controls the processor sampling. The program accepts heading and initialization parameters provided by the operator, performs a fast I/O handshake to acquire the raw data from the LDV-A/D computer interface, and writes the raw data and initialization parameters onto floppy-disk. In addition, the data acquisition program has a limited capability for on-line data reduction so that key results may be monitored during a run. The off-line data reduction program converts the raw data into instantaneous velocities and computes all the statistical quantities. The off-line program also has options to plot histograms of the raw data, to plot profiles of the reduced data, and to filter out noise.

3.1 Data Acquisition Program

The data acquisition program structure can be divided into four areas: (1) initialization of variables, (2) data acquisition, (3) limited data reduction, and (4) raw data storage (see the block diagram in Fig. 8). During the initialization stage of the program, relevant test parameters are input to be saved along with the data. These parameters include the spatial coordinates, Bragg and mixing frequencies, and angle of the third beam pair. Fringe spacings and probe dimensions are also calculated at this time.

Data is acquired from the LDV computer interface by means of a fast I/O handshake. The operator requests that a certain number of data samples be taken and stored in the data buffer as described in Section 2.6. The 16-bit data words are put into an integer array, three words wide, containing values for the three velocity components. The number acquired from the counter occupies only bits 0-3 of the data word. Bits 10-13 specify the range, and bits 14 and 15 are unused. The data array, along with the initialization parameters given by the operator, are stored in a floppy-disk file. As an option some of the data can be reduced on-line, before the raw data is written

to floppy-disk. This on-line reduction feature uses the same algorithms as the off-line data reduction program, but is faster since it computes fewer quantities. Data acquisition times depend on many factors other than the software, but on-line data reduction takes about 30 seconds per point on the HP 9845B, and each file write to the floppy-disk requires about 20 seconds, assuming 2000 samples are taken per point.

3.2 Data Reduction Program

A separate off-line program is used to reduce the complete data. The data reduction program has four major sections: (1) data read from floppy-disk, (2) conversion of raw data to instantaneous velocities, (3) calculation of sums of instantaneous velocity components, and (4) calculation of moments from the sums (see the block diagram in Fig. 9). Since the HP 9845B is a relatively slow micro-computer, the bulk of the calculation routines are written in ASSEMBLY language to reduce computation time.

The program reads the initialization parameters and raw data which were written on floppy-disk by the data acquisition program. The raw data is used together with the mixing frequencies, Bragg shift frequency, third beam crossing angle, and fringe spacings to calculate the three velocity components for each sample. The velocities for each sample are calculated in turn, and a running sum is maintained. After the samples have been summed, the average values of the various moments are computed, and the mean velocities and turbulence quantities are calculated from them. Reading from the floppy-disk takes about 20 seconds, while the data reduction including histogram plots requires about 90 seconds per point on the HP 9845B.

The procedure for data reduction is fairly straightforward. The raw data is converted to frequencies using the range set on the processor. The measured frequency is given by the relation

$$f_{res} = 3.2 * 10^4 / 2^R * D \quad (2)$$

where D = raw data.

The velocities are then calculated from the resultant frequencies, by the relation:

$$velocity = d_f * ((f_{bragg} - f_{mix}) - f_{res}) \quad (3)$$

Once the velocities have been evaluated, average values of the various moments are computed as defined below:

$$\begin{aligned} S_u &= S\langle u(i) \rangle / N_s \\ S_v &= S\langle v(i) \rangle / N_s \\ S_w &= S\langle w(i) \rangle / N_s \\ S_{uu} &= S\langle u(i)u(i) \rangle / N_s \\ S_{uv} &= S\langle u(i)v(i) \rangle / N_s \\ S_{uw} &= S\langle u(i)w(i) \rangle / N_s \\ S_{vv} &= S\langle v(i)v(i) \rangle / N_s \\ S_{vw} &= S\langle v(i)w(i) \rangle / N_s \\ S_{ww} &= S\langle w(i)w(i) \rangle / N_s \\ S_{uuv} &= S\langle u(i)u(i)v(i) \rangle / N_s \\ S_{uuw} &= S\langle u(i)u(i)w(i) \rangle / N_s \\ S_{uvv} &= S\langle u(i)v(i)v(i) \rangle / N_s \\ S_{uww} &= S\langle u(i)w(i)w(i) \rangle / N_s \\ S_{uvw} &= S\langle v(i)v(i)w(i) \rangle / N_s \end{aligned} \quad (4)$$

Using these definitions, the signal statistics are then calculated assuming nearly infinite sample size:

$$\begin{aligned} \bar{u} &= S_u \\ \bar{v} &= S_v \\ \bar{w} &= S_w \\ \overline{u'^2} &= S_{uu} - S_u S_u \\ \overline{v'^2} &= S_{vv} - S_v S_v \end{aligned} \quad (5)$$

$$\begin{aligned}
\overline{w'^2} &= Sww - SwSw \\
\overline{u'v'} &= Suv - SuSv \\
\overline{u'w'} &= Suw - SuSw \\
\overline{v'w'} &= SvW - SvSw \\
\overline{u'^2v'} &= Suuv - 2SuSuv - SvSuu + 2SvSuSu \\
\overline{u'^2w'} &= Suuw - 2SuSuw - SwSuu + 2SwSuSu \\
\overline{u'v'^2} &= Suvv - 2SvSuv - SuSvv + 2SuSvSv \\
\overline{u'w'^2} &= Suww - 2SwSuw - SuSwW + 2SuSwSw \\
\overline{u'v'w'} &= Suvw - SuSvW - SvSuw - SwSuv + 2SuSvSw
\end{aligned}$$

The implementation of the data reduction software is somewhat more complex. In order to achieve reasonable running times, the ASSEMBLY code is optimized for speed rather than clarity of operation. Operations which do not change between samples are performed only once. Thus,

$$\text{velocity} = d_f * ((f_{\text{bragg}} - f_{\text{mix}}) - f_{\text{res}}) * 10^6 \quad (6)$$

becomes

$$f_{\text{int}} = f_{\text{bragg}} - f_{\text{mix}} ; \quad d_{\text{int}} = d_f * 10^6 \quad (7)$$

$$\text{hence, } \text{velocity} = d_{\text{int}} * (f_{\text{int}} - f_{\text{res}}) \quad (8)$$

Additional calculations are required to obtain the w component velocity. Since the third set of beams measures $w_v = w \sin 45^\circ + v \cos 45^\circ$, to find w, we must also perform the calculation:

$$w = (w_v - v \cos 45^\circ) / \sin 45^\circ. \quad (9)$$

The raw data output of the digitizer is in the form of a 10 bit data word, giving a range of possible values (counts) from 0 to 1023. Since 2000 samples are normally taken for each point, a given value may be encountered many times. Accordingly, each time a new value of the raw data is encountered, the corresponding velocity is calculated and stored in a look-up table. The next time that value is encountered by the program, the

proper velocity can be easily looked up, eliminating the need for another time-consuming real variable calculation. The exact running procedure of the data reduction program therefore, is as follows:

1. Reads header and all raw data for a particular point from floppy disk.
2. Strips bits 10-13 from three raw data words corresponding to the three different velocity components, and uses them to calculate the three ranges set on the A/D.
3. Calculates various intermediate values which remain the same throughout the point.
4. Reads a data word, strips off bits 0-9, and uses this raw datum as an index to look up its corresponding velocity.
5. If it is a new value, the program calculates the velocity using the raw datum, and stores it in the appropriate place in the look-up table.
6. For the w velocity component, it finds the actual velocity from $w = (w_v - v \cos 45^\circ) / \sin 45^\circ$.
7. Performs steps 4, 5, and 6 three times--once for each velocity component of the sample.
8. Updates the running sums of the velocity components and products of the velocity components.
9. Performs steps 4 through 8 for all samples.
10. Uses the sums to obtain the average velocities, Reynolds stresses, and third order products, and then prints these quantities.

- 11 Plots histograms of the raw data for all three channels.
12. Performs steps 1 through 11 for each profile point.
13. Plots profiles of the reduced data.
14. Tabulates normalized data profiles.
15. Writes a summary file containing the reduced data to disk.

The data reduction program also has a routine to filter noise and spurious data. Each of the 2000 data samples has 3 counts corresponding to the 3 velocities associated with it. The filtering routine causes the data reduction program to ignore samples associated with counts which are excessively far from the mean.

The filtering routine first sorts the data into three frequency tables, one for each channel. In a table, each count, i , has associated with it a number, S_{ij} , which is the number of samples with that particular count.

The filtering routine finds the average count for each channel by going through the frequency tables and using the formula

$$C_j = \frac{\sum_{i=1}^{1024} i \times S_{ij}}{N_s} \quad (10)$$

where $j = 1, 2, 3$ = channel number

C_j = average count

N_s = number of samples.

Next, the filtering routine finds the standard deviation by performing the summation

$$\sigma_j^2 = \frac{\sum_{i=1}^{1024} S_{ij} (i - C_j)^2}{N_s} \quad (11)$$

where σ_j = standard deviation.

An input variable called S_{dev} is read by the filtering routine. All counts further than S_{dev} standard deviations away from the mean will be filtered out. This is done by multiplying S_{dev} by σ_j for each channel, and going through the frequency tables one more time. If the magnitude of $(i - C_j) > S_{dev} * \sigma_j$, then S_{ij} is set equal to zero.

The routine which calculates the statistical quantities takes each sample one at a time. The three counts associated with each sample are found and looked up in the frequency table. If for any count the table entry (S_{ij}) is zero, then the sample is discarded. Thus any sample which is excessively far from the mean in any one of its three counts is not used.

4. OPERATING PROCEDURE

4.1 Alignment Procedure

To achieve the best possible beam crossing and the most effective measuring volume, each module in the optics system must be carefully aligned. Detailed alignment instructions for individual optical components are given in Ref. 3. The overall alignment procedure is described in this report. Component numbers in this section refer to those shown in Fig. 1.

The first step is to check that the laser output beam is parallel to the optics table at the specified height, using the system alignment blocks. The collimator (1) should be positioned so that the laser beam goes through the center of the lens, and focused so that the beam waist occurs at the cross-over point. The aligned beam then passes through the polarization rotator (2) and into the color separator box. The attenuator (3) ensures that the beam has horizontal polarity at this point.

Each component of the color separator must be aligned separately. The dispersion prism (4) should be aligned so that the path length is the same in both prisms and is parallel to the

base of each prism. Several beams emerge from the prism, with the two brightest ones being green (514.5 nm) and blue (488 nm). The two beams are reflected out of the box using mirrors. The beams should be centered on mirrors (6) and (7). As the beams come out of the box, alignment blocks are used to check the beam positions. If both beams are off-axis in the same direction, mirror (5) is used to align them. If only one beam is off, the appropriate mirror, (6) or (7), is used for the adjustment. Beam splitter efficiency is maximized when the beam polarization is perpendicular to the plane of the split beams. This is achieved through components (8) and (11). The beam splitters (9) and (12) cannot split the beam intensity exactly in half, so one beam of each pair is always slightly brighter. Since the Bragg cells (13) and (14) normally attenuate the shifted beam, the brightest beam is frequency shifted so that the output beam pairs have nearly equal intensities. Detailed instructions for aligning the Bragg cells are given in Ref. 4. After passing through the Bragg cells, directional wedges are used to project the beams onto a distant surface (~ 3 m). Any beam misalignments are more easily seen this way, and with the use of a marked mask, the beams can be adjusted to the correct orientation. Mirrors (6, 7) are used to adjust the unshifted beams, and the beam steering modules (15) are used for rotating the shifted beams along two circular arcs.

The next check is to ensure that the beams are parallel to each leg of the traverse mechanism on the transmitting optics table, so that beam alignment is maintained while traversing. The dichroic filter (Fig. 2) is adjusted so that the green beam pairs are split equally. Once the beams pass through the transmitting optics, the four beams must be arranged so that they all cross at the same position. A microscope objective is used to view the beam crossing. If the four beams are not symmetric about the optical axis, mirrors (6) or (7) can be adjusted to correct this. The beam steering modules (15) are used to ensure that the beams cross at the same point along the axis. The

third pair of beams are now aligned (by eye) so that they also cross at the same point as the main line beams.

To align the receiving optics, a piece of translucent tape is placed at the beam intersection point to scatter the laser light. By tracking the scattered light, the receiving optics are aligned to give a sharp image at the pin-hole aperture in front of the photomultiplier tubes.

4.2 Signal Processing

The sensitivity of the photomultiplier (PM) tubes used in the receiving optics can be varied by varying the applied voltage. Typically, a voltage of about 1000 volts is applied. This voltage can be increased to make the PM tubes more sensitive, as long as the threshold levels on the processors are increased accordingly, since the amount of noise picked up is also increased.

The measured signal is mixed electronically with the signal from a sine wave generator. The frequency of the sine wave (the mixing frequency) is chosen based on the expected flow velocities. The mixing frequency is chosen such that the difference between it and the Bragg shift is about twice the maximum expected Doppler frequency. This allows enough margin for fluctuations about the expected Doppler frequency and still have a remaining nonzero resultant frequency ($f_{res} = f_{bragg} - f_{mix} - f_{dopp}$). If f_{mix} is too high, a biasing results, similar to the fringe biasing caused by stationary fringes. The number of fringes crossed by a particle per second (as seen by the processor) is proportional to $\Delta t(f_{bragg} - f_{mix})$, where Δt is the time taken by a particle to cross one fringe. Noting that Δt is only determined by the fringe spacing and the flow speed, if f_{mix} is increased, the number of fringes crossed by a particle is effectively reduced. This means that signals from particles which cross the fringes at an angle may not have enough fringe crossings (8) to be validated, and hence a bias towards particles moving perpendicularly to

the fringes (higher velocity) results. This gives a higher mean velocity but a lower fluctuation level.

The amplitude of the sine wave must be chosen so that an adequate signal-to-noise ratio is maintained. Typically, a peak-to-peak amplitude of 1 volt is required.

The mixed signal is then fed into single particle burst (Macrodyne) processors. The high pass and low pass filter frequencies are set so that the processor frequency is centered between the two. Usually the high pass filter is set at 0.5 Mhz, which is the lowest non-zero setting, and the low pass filter is set anywhere from 2 to 16 Mhz, depending on the magnitude of the velocity component being measured. The filter bandwidth should be broad enough so that no parts of the fluctuating signal are attenuated.

The gain on the processor is normally set to 10. The output of the processors is displayed on an oscilloscope, and the Doppler signals should read about 1 volt peak-to-peak. The PM tube voltage can be adjusted so that the signal is at the desired level. Signal levels of more than about 1 volt end up being clipped and will therefore not be validated by the processor.

The comparator accuracy for the 5/8 signal validation test can be set between 0 and 10 count variation. The 0 setting is the most accurate, and 11 is off. The processor manufacturer recommends that this level be usually set to 9.

The range on the processor sets the bandwidth of frequencies that the processor can see, according to Table 1. For each range, a particular frequency corresponds to a count, from 0 to 1023. The processor frequency should be matched to a number in the central column in Table 1, and the corresponding range should be set on the Macrodyne processor. An additional check can be made on the range setting by monitoring the analog output from the Macrodyne on a DC voltmeter. The output ranges from 0 to 10 volts, which corresponds to the counts from 0 to 1023. The correct range setting is that which gives about 5 volts on the voltmeter at the operating velocity.

The threshold is then set for the multi-sequence check. The threshold should be set so that the data rate is about half of the data rate at zero threshold. Another check is to block one beam in each color pair. The data rate on the corresponding channel should be zero when one beam is blocked.

To obtain good data rates in air, the flow must be seeded with uniformly sized particles. Smoke, obtained from burning mineral oil or incense, provides particles of approximately 2 μm .

The computer interface should be set up as shown in Fig. 10. First choose the coincidence mode. If shear stresses are to be evaluated, coincidence on all 3 channels must be selected. The coincidence time (5 to 50 μs) sets the window width during which coincidence is defined. This should be set at 5 μs . If only the individual velocities are desired, then the random mode can be selected. The dead time (5 to 50 μs) should then be set so that data from one particle is not recorded twice. A setting of about 25 μs is recommended for low-speed flows.

The number of words that must be multiplexed can be calculated as follows: $\# \text{words} = \# \text{inputs} + 2$. This number should be rounded to the nearest even number. In the present case a setting of six is used. The event mode is set to LDV (digital data only). The LDV-A/D switch enables both digital and analog data to be interfaced simultaneously. The counter clock frequency is set to equal the approximate data rate, and the computer select is set to HP.

5. INHERENT PROBLEMS

Some design problems inherent to 3-component LDV systems are discussed in this section. One main problem with some of the earlier designs which called for splitting a color to create the third beam pair had to do with cross-talk. This is where signals from the two channels bearing the same color could not be adequately separated. In the original design of the present

set-up, the polarization of one of the green beam pairs was rotated relative to the other, so that the receiving optics could distinguish between the two signals. However, using a relatively large angle for the off-axis beams (45°) and two separate collection lenses, cross-talk between the two green channels has been almost eliminated, thus making the polarization rotation dispensable.

Some earlier designs of 3-component LDV systems measured the $u + w$ velocity component with the off-axis third beam pair. It is shown in Ref. 1 that measuring the $v + w$ component instead, as done in the present system, reduces the uncertainty in the w component relative to the uncertainty in w from these earlier systems.

Another problem has to do with signal coincidence. Details of the probe volumes for the present system are shown in Fig. 3. It is clearly illustrated how the cross-over region between the three sets of beams forms a very small fraction of the overall probe volume. Thus, with heavy seeding (necessary for three-channel work), the electronics may validate data received from different particles which are not necessarily in the cross-over region but are within the coincidence time window set on the interface. This results in a lack of correlation between the measured velocities, and causes the evaluated shear stresses to be inaccurate. Two schemes have been used in an attempt to minimize this problem. First, the coincidence time was made so short that measurements from different particles may be considered instantaneous. The minimum setting of $5 \mu s$ available in the present hardware was used; this is equivalent to about half the flight time of a particle passing through the probe volume. The second procedure involved reducing the effective "viewed" probe length and thereby reducing the probability of this "apparent coincidence" (Fig. 5). A collection angle θ of 30° and an aperture diameter of 0.5 mm were used to give an effective length of about 1.5 mm.

6. SAMPLE RESULTS

As an initial check on the accuracy of the present system, some preliminary measurements have been made in a vortex/mixing layer interaction, previously investigated using the cross-wire technique (Ref. 6). Since the induced cross-flow angles in this interaction are only 5° - 10° , cross-wire measurements are expected to be accurate to within about 5%. A schematic of the experimental set-up is shown in Figs. 11a and b. LDV measurements of the secondary flow velocities at one streamwise station ($x = 229$ mm) are presented and compared to the cross-wire measurements in Figs. 12a and b. LDV and cross-wire measurements of the turbulence quantities at one spanwise position ($z = 13$ mm) are compared in Fig. 13.

The secondary flow velocities are qualitatively similar, although the LDV measurements indicate a somewhat higher \bar{w} . The normal intensity $\overline{w'^2}$ also seems slightly high. The higher \bar{w} measurements are more likely caused by a slight misalignment of the beams relative to the tunnel axis rather than by remnants of the apparent coincidence problem (discussed above in Section 5), since the latter would not affect the \bar{w} measurements. $\overline{v'^2}$ seems to agree very well whereas $\overline{u'^2}$ is a bit low, and since \bar{u} was a bit high, this was probably a result of fringe biasing caused by too high a mixing frequency (as discussed above in Section 4.2). (The mixing frequencies used for these measurements were 37.5 MHz for the u channel, and 38 MHz for the v and w_v channels.) However, the normal stress measurements agree to within 10%, and the shear stresses are consequently affected. $\overline{u'v'}$ is somewhat low (about 20%) whereas $\overline{u'w'}$ is low by almost a factor of two. The measurement of $\overline{u'w'}$ with the present system seems to be very sensitive. This is due to the fact the $w_v * u$ is of the same order as $u * v \cos 45^\circ$, so any small error in the measurement of these velocities or the off-axis angle can result in large errors in $\overline{u'w'}$. $\overline{v'w'}$ in vortex affected flows is generally of the same order as $\overline{u'w'}$ and this seems to be the case with

the present measurements. With a cross-wire, $\overline{v'w'}$ has to be evaluated from measurements made in four different planes about the probe axis, and hence was not measured here. The comparisons clearly demonstrate the potential of the system in measuring detailed mean flow and turbulence quantities in three-dimensional flows. Work is in progress on optimizing the problems discussed above so that the measurement accuracies may be improved.

6. CONCLUDING REMARKS

A 3-component LDV system, capable of measuring all three components of velocity simultaneously has been developed for use in low-speed three-dimensional flows. All the six components of Reynolds shear stress and higher order products of interest can hence be evaluated. The approach followed was to convert an existing 2-component system by splitting one of the colors to produce the third beam pair. The additional optical hardware required for this process is relatively minor.

For the first time, three-component measurements made with an LDV system have been compared directly with those obtained with the cross-wire technique, in a three-dimensional flow field where both techniques are expected to perform satisfactorily. The preliminary measurements are encouraging and work is in progress on improving the system accuracy.

APPENDIX

SOFTWARE FOR THE HP 9845B DESK-TOP COMPUTER

Complete listings of two programs written in BASIC and ASSEMBLY languages are included in this appendix: "LDV" for data acquisition, some on-line data reduction, and storage of data on floppy disks; and "STAT" for complete off-line data reduction from files written to disk.

```

10  REM PROGRAM LDV
20  ! PROGRAM TO ACQUIRE DATA FROM THREE-COMPONENT LDV SYSTEM
30  ! The program asks for initialization data and calculates the
40  ! calibration constants from them. It reads 3 channels of raw
50  ! LV data from the LDV A/D CI and writes them to a disk file
60  ! together with the calibration constants. The program can reduce
70  ! the data and display real-time histograms of the raw LV data
80  ! if desired.
90  OPTION BASE 1
100 ICOM 16600
110 IDELETE ALL
120 IASSEMBLE Find_vel
130 IASSEMBLE Data_trans
140 IASSEMBLE Draw_hist
150 COM INTEGER Data(3,2000),D1(2000,6),Ns,Nn 'D1 is data buffer
160 COM REAL Df1,Df2,Df3
170 COM REAL Theta,Nub,Num1x1,Num1x2,Num1x3
180 COM REAL Su,Sv,Sw,Suu,Suv,Suw,Suu,Suv,Suw
190 DIM Date$(80),File$(6),Name$(4),Titl$(80)
200 REAL Xpos,Ypos,Zpos,Pos(30,3)
210 INTEGER A,B,Range1,Range2
220 INTEGER Run,Dn,Nss,N
230 REAL Ph1,D,F,Ph2
240 REAL Lam1,Lam2,Lam3,Db,Prwid1,Prlen1
250 REAL Re,Ue
260 REAL Prwid2,Prwid3,Prlen2,Prlen3
270 REAL Nfr1,Nfr2,Nfr3
280 REAL U,V,W
290 Ns=2000
300 DEG
310 PRINT
320 PRINT " ** << PROGRAM LDV : 3-COMPONENT VELOCITY DATA >> ** "
330 PRINT
340 PRINT "PROGRAM STRUCTURE"
350 PRINT " 1. INITIALIZE VARIABLES AND CALCULATE PARAMETERS"
360 PRINT " 2. ACQUIRE DATA FROM A/D"
370 PRINT " 3. WRITE TO FLOPPY DISC"
380 PRINT
390 !
400 ! ** CHECK HISTOGRAMS **
410 !
420 Ans$="N"
430 INPUT "DO YOU WISH TO LOOK AT HISTOGRAMS ? (Y/N, DEFAULT N)",Ans$
440 IF Ans$="Y" THEN GOSUB Hist
450 !
460 ! ** INITIALIZE RUN **
470 !
480 PRINT " ** INITIALIZATION ** "
490 Run=1
500 PRINT " ENTER RUN PARAMETERS:"
510 PRINT
520 INPUT "Enter date and time:",Date$
530 INPUT "Enter 1-Line Name For Profile :",Titl$
540 INPUT "No. of data samples per point (2000 maxm.) :",Ns
550 !
560 ! BEAM SPACING J IS FIXED
570 !
580 D=.013
590 !
600 ! FOCAL LENGTH F IS FIXED

```

```

610 |
620 F=.381
630 |
640 | WAVELENGTHS OF 3 BEAMS ARE FIXED
650 |
660 Lam1=4.88E-7
670 Lam2=5.145E-7
680 Lam3=5.145E-7
690 |
700 | REFERENCE VELOCITY SET TO ZERO
710 |
720 Ue=0
730 |
740 | CALCULATE HALF-ANGLES FROM BEAM SPACINGS AND FOCAL LENGTH
750 |
760 Ph1=ATN(D/2/F)
770 Ph2=Ph1*2
780 |
790 | CALCULATE FRINGE SPACINGS FROM WAVELENGTHS AND HALF-ANGLE
800 |
810 Df1=Lam1/2/SIN(Ph1)
820 Df2=Lam2/2/SIN(Ph1)
830 Df3=Lam3/2/SIN(Ph1)
840 |
850 | BEAM DIAMETER Db IS FIXED
860 |
870 Db=1.20E-3
880 |
890 | CALCULATE PROBE VOLUME WIDTH AND LENGTH
900 |
910 Prwid1=4*Lam1*F/PI/Db/COS(Ph1)
920 Prlen1=4*Lam1*F/PI/Db/SIN(Ph1)
930 Prwid2=4*Lam2*F/PI/Db/COS(Ph1)
940 Prlen2=4*Lam2*F/PI/Db/SIN(Ph1)
950 Prwid3=Prwid2
960 Prlen3=Prlen2
970 Nfr1=Prwid1/Df1
980 Nfr2=Prwid2/Df2
990 Nfr3=Nfr2
1000 |
1010 | GET MORE RUN PARAMETERS
1020 |
1030 INPUT "Enter Bragg shift frequency (MHz) :",Nub
1040 Kill_w=3
1050 INPUT "Is this a two-channel or three-channel run (2/3, default 3) ?",Kill_w
1060 IF Kill_w=3 THEN GOTO 1110
1070 INPUT "Enter mixing frequency (MHz, 2 nos.)",Numix1,Numix2
1080 Numix3=0
1090 Theta=0
1100 GOTO 1130
1110 INPUT "Enter mixing frequency (MHz, 3 nos.)",Numix1,Numix2,Numix3
1120 INPUT "Enter 3rd beam angle (degrs) :",Theta
1130 INPUT "Enter tunnel reference voltage (volts)",Vref
1140 PRINTER IS 0
1150 |
1160 | PRINT HEADER
1170 |
1180 PRINT Title$
1190 PRINT "TEST DATE AND TIME :",Date$
1200 PRINT "BEAM SPACINGS (m) =",D

```



```

1210 PRINT "TOTAL ANGLE BETWEEN BEAMS (Degrs) =",Ph2
1220 PRINT "FRINGE SPACINGS (m) =",Df1,Df2,Df3
1230 PRINT "PROBE WIDTHS (m) =",Prwid1,Prwid2,Prwid3
1240 PRINT "PROBE LENGTHS (m) =",Prlen1,Prlen2,Prlen3
1250 PRINT "NO. OF FRINGES = ",Nfr1,Nfr2,Nfr3
1260 PRINT "BRAGG SHIFT FREQUENCY (MHz) =",Nub
1270 PRINT "MIXING FREQUENCY (MHz) =",Numix1,Numix2,Numix3
1280 PRINT "THIRD BEAM SET ANGLE (Degrs) =",Thet3
1290 PRINT "RUN NUMBER = ",Run
1300 PRINT "TUNNEL REFERENCE VOLTAGE (volts) =",Vref
1310 PRINTER IS 16
1320 Ans$="N"
1330 INPUT "DO YOU WISH TO MAKE ANY CHANGES? (Y/N, DEFAULT N)",Ans$
1340 IF Ans$="Y" THEN GOTO 540
1350 !
1360 ! DATA ACQUISITION
1370 !
1380 PRINT
1390 PRINT " ** DATA ACQUISITION ** "
1400 REDIM Data(3,Ns)
1410 Ans$="N"
1420 INPUT "DO YOU WISH TO CHANGE FILE NAME ? (Y/N, DEFAULT N)",Ans$
1430 IF Ans$="N" THEN GOTO 1490
1440 INPUT "Enter 4-digit filename:",Name$
1450 PRINTER IS 0
1460 PRINT "4-digit filename for profile :",Name$
1470 INPUT "Enter Disk Number :",Dn
1480 PRINT "Disk Number =",Dn
1490 PRINTER IS 16
1500 INPUT "Enter X, Y, AND Z locations:",Xpos,Ypos,Zpos
1510 !
1520 ! TAKE DATA
1530 !
1540 PRINTER IS 0
1550 PRINT
1560 PRINT
1570 PRINT "POINT NUMBER IN PROFILE : ",Run
1580 PRINT "X,Y,Z = ",Xpos,Ypos,Zpos
1590 PRINTER IS 16
1600 GOSUB Atod
1610 !
1620 ! CALCULATE SAMPLE VELOCITIES
1630 !
1640 Ans$="N"
1650 INPUT "DO YOU WISH TO OBTAIN ESTIMATES OF U AND V ? (Y/N, DEFAULT N)",Ans$
1660 IF Ans$="N" THEN GOTO 1720
1670 RAD
1680 ICALL Find_vel
1690 GOTO 2420
1700 !
1710 ! WRITE CALIBRATION CONSTANTS AND DATA TO DISK
1720 !
1730 GOSUB Dfile
1740 Ans$="Y"
1750 INPUT "DO YOU WISH TO TAKE ANOTHER POINT ? (Y/N, DEFAULT Y)",Ans$
1760 IF Ans$="N" THEN GOTO 1820
1770 Run=Run+1
1780 Ans$="N"
1790 INPUT "DO YOU WISH TO CHANGE ANY PARAMETERS ? (Y/N, DEFAULT N)",Ans$
1800 IF Ans$="Y" THEN GOTO 540

```



```

2410 Uguess:      ! Estimate U-component of velocity
2420 DISP "CALCULATING ESTIMATES OF U AND V VELOCITIES"
2430 !
2440 !
2450 IF Kill_u=3 THEN GOTO 2490
2460 DATA 0,0,0
2470 RESTORE 2460
2480 READ Su,Suu,Suw
2490 Ubar=Su/Ns
2500 Vbar=Su/Ns
2510 Wbar=Su/Ns
2520 Upri2=Suu/Ns-Ubar*Ubar
2530 Vpri2=Suv/Ns-Vbar*Vbar
2540 Wpri2=Sww/Ns-Wbar*Wbar
2550 Uvbar=Suv/Ns-Ubar*Vbar
2560 Uwbar=Suw/Ns-Ubar*Wbar
2570 Vwbar=Sww/Ns-Vbar*Wbar
2580 PRINTER IS 0
2590 PRINT "ESTIMATE OF QUANTITIES FROM SAMPLES :",Ns
2600 PRINT "Ubar =",Ubar
2610 PRINT "Vbar =",Vbar
2620 PRINT "Wbar =",Wbar
2630 PRINT "Upri2 =",Upri2
2640 PRINT "Vpri2 =",Vpri2
2650 PRINT "Wpri2 =",Wpri2
2660 PRINT "Uvbar =",Uvbar
2670 PRINT "Uwbar =",Uwbar
2680 PRINT "Vwbar =",Vwbar
2690 GOTO 1700
2700 ISOURCE      NAM Find_vel
2710 ISOURCE !
2720 ISOURCE ! This subroutine converts raw data counts into instantaneous
2730 ISOURCE ! velocities, then sums several different products of the
2740 ISOURCE ! velocity components. All input and output data is passed
2750 ISOURCE ! through the COMMON storage area. The inputs are the raw
2760 ISOURCE ! data array (Arrayd), the Bragg shift frequency (Nub), the
2770 ISOURCE ! mixing frequencies (Numix1, Numix2, Num x3), the fringe
2780 ISOURCE ! spacings (Df1, Df2, Df3), the crossing angle of the third
2790 ISOURCE ! beam (Theta), and the number of samples in a data point (Ns).
2800 ISOURCE ! The outputs are the summations of various products of the
2810 ISOURCE ! velocity components, including U, V, W, U*U, V*V, W*W, U*V,
2820 ISOURCE ! U*W, and V*W.
2830 ISOURCE !
2840 ISOURCE      EXT Get_value      ! Declare subroutines stored
2850 ISOURCE      EXT Get_info      ! outside of the main program.
2860 ISOURCE      EXT Get_element
2870 ISOURCE      EXT Int_to_rel
2880 ISOURCE      EXT Rel_math
2890 ISOURCE      EXT Put_value
2900 ISOURCE !
2910 ISOURCE      COM
2920 ISOURCE Data_par: INT (*)      ! Declare common variables.
2930 ISOURCE D1_par:  INT (*)
2940 ISOURCE Ns_par:   INT
2950 ISOURCE Nn_par:   INT
2960 ISOURCE Df1_par:  REL
2970 ISOURCE Df2_par:  REL
2980 ISOURCE Df3_par:  REL
2990 ISOURCE Theta_par: REL
3000 ISOURCE Nub_par:  REL

```

```

3010 ISOURCE Nm1x1_par: REL
3020 ISOURCE Nm1x2_par: REL
3030 ISOURCE Nm1x3_par: REL
3040 ISOURCE Su_par: REL
3050 ISOURCE Sv_par: REL
3060 ISOURCE Sw_par: REL
3070 ISOURCE Suu_par: REL
3080 ISOURCE Svv_par: REL
3090 ISOURCE Swv_par: REL
3100 ISOURCE Suu_par: REL
3110 ISOURCE Swv_par: REL
3120 ISOURCE Svv_par: REL
3130 ISOURCE !
3140 ISOURCE Arrayd: BSS 39      ! Reserve space for data array
3150 ISOURCE Elementd: EQU Arrayd+16 ! descriptor.
3160 ISOURCE Array1: BSS 4096    ! Reserve space for lookup tables used
3170 ISOURCE Array2: BSS 4096    ! for count to velocity conversion.
3180 ISOURCE Array3: BSS 4096
3190 ISOURCE Ns: BSS 1          ! Reserve space for various input
3200 ISOURCE Df1: BSS 4         ! and output variables.
3210 ISOURCE Df2: BSS 4
3220 ISOURCE Df3: BSS 4
3230 ISOURCE Theta: BSS 4
3240 ISOURCE Nub: BSS 4
3250 ISOURCE Num1x1: BSS 4
3260 ISOURCE Num1x2: BSS 4
3270 ISOURCE Num1x3: BSS 4
3280 ISOURCE Su: BSS 4
3290 ISOURCE Sv: BSS 4
3300 ISOURCE Sw: BSS 4
3310 ISOURCE Suu: BSS 4
3320 ISOURCE Svv: BSS 4
3330 ISOURCE Swv: BSS 4
3340 ISOURCE Suu: BSS 4
3350 ISOURCE Swv: BSS 4
3360 ISOURCE Svv: BSS 4
3370 ISOURCE Count: BSS 1      ! Count and I are general purpose index vari-
3380 ISOURCE I: BSS 1          ! ables. Count is usually 0, 1, or 2, to denote
3390 ISOURCE Check: BSS 1      ! whether U,V, or W is being calculated.
3400 ISOURCE Int: BSS 1        ! Int, Address, and Offset are all general
3410 ISOURCE Address: BSS 1    ! purpose storage areas.
3420 ISOURCE Offset: BSS 1
3430 ISOURCE R1: BSS 4         ! R1, R2, and R3 are the count-to-
3440 ISOURCE R2: BSS 4         ! frequency conversion factors.
3450 ISOURCE R3: BSS 4
3460 ISOURCE Xvar: BSS 4      ! Xvar and Yvar are general purpose real
3470 ISOURCE Yvar: BSS 4      ! number storage areas.
3480 ISOURCE U: BSS 4         ! reserve space for instantaneous velocity
3490 ISOURCE V: BSS 4         ! components.
3500 ISOURCE W: BSS 4
3510 ISOURCE Uu: BSS 4
3520 ISOURCE Uv: BSS 4
3530 ISOURCE Cos: BSS 4       ! Cos and Sin are the cosine and sine of Theta
3540 ISOURCE Sin: BSS 4
3550 ISOURCE Rad: DAT 5.729578E1
3560 ISOURCE Mill: DAT 1.E6
3570 ISOURCE One: DAT 1.
3580 ISOURCE Zero: DAT 0.
3590 ISOURCE !
3600 ISOURCE SUB

```

```

3610 ISOURCE Find_vel: LDA =Ns          ! Get number of samples.
3620 ISOURCE          LDB =Ns_par
3630 ISOURCE          JSM Get_value
3640 ISOURCE          LDA =Arrayd       ! Get parameters of data array.
3650 ISOURCE          LDB =Data_par
3660 ISOURCE          JSM Get_info
3670 ISOURCE          LDA =Df1         ! Get input parameters.
3680 ISOURCE          LDB =Df1_par
3690 ISOURCE          JSM Get_value
3700 ISOURCE          LDA =Df2
3710 ISOURCE          LDB =Df2_par
3720 ISOURCE          JSM Get_value
3730 ISOURCE          LDA =Df3
3740 ISOURCE          LDB =Df3_par
3750 ISOURCE          JSM Get_value
3760 ISOURCE          LDA =Theta
3770 ISOURCE          LDB =Theta_par
3780 ISOURCE          JSM Get_value
3790 ISOURCE          LDA =Nub
3800 ISOURCE          LDB =Nub_par
3810 ISOURCE          JSM Get_value
3820 ISOURCE          LDA =Numix1
3830 ISOURCE          LDB =Nmix1_par
3840 ISOURCE          JSM Get_value
3850 ISOURCE          LDA =Numix2
3860 ISOURCE          LDB =Nmix2_par
3870 ISOURCE          JSM Get_value
3880 ISOURCE          LDA =Numix3
3890 ISOURCE          LDB =Nmix3_par
3900 ISOURCE          JSM Get_value
3910 ISOURCE          !
3920 ISOURCE          ! The loop headed by Get_freq is repeated three times to get
3930 ISOURCE          ! the count-to-frequency conversion factors (which depend on
3940 ISOURCE          ! the range) for U, V, and W. Whenever a loop is controlled by
3950 ISOURCE          ! the variable "Count", the loop contains operations which are
3960 ISOURCE          ! the same for U, V, and W.
3970 ISOURCE          !
3980 ISOURCE          LDA =0
3990 ISOURCE          STA Count
4000 ISOURCE          Get_freq: LDA Cou          ! Get the first word of the column of
4010 ISOURCE          LDB Ns          ! the data array which contains the
4020 ISOURCE          MPY          ! velocity component for which we want
4030 ISOURCE          STA Elementd    ! to get the range.
4040 ISOURCE          LDA =Int
4050 ISOURCE          LDB =Arrayd
4060 ISOURCE          JSM Get_element
4070 ISOURCE          LDA Int          ! Mask and rotate to get the four
4080 ISOURCE          LDB =15360      ! bits containing the range.
4090 ISOURCE          AND B
4100 ISOURCE          SAR 10
4110 ISOURCE          TCA          ! Subtract from 15 to get the
4120 ISOURCE          LDB =15        ! actual range.
4130 ISOURCE          ADA B
4140 ISOURCE          LDB =1
4150 ISOURCE          SZA Loopend
4160 ISOURCE          Loop: SBL 1      ! Use the range to find the power
4170 ISOURCE          DSZ A          ! of two needed for the divisor.
4180 ISOURCE          JMP Loop
4190 ISOURCE          Loopend: STB Int
4200 ISOURCE          LDA =Int      ! Convert the power of two into a

```

4210	ISOURCE	STA Oper_1	real number.
4220	ISOURCE	LDA =Yvar	
4230	ISOURCE	STA Result	
4240	ISOURCE	JSM Int_to_rel	
4250	ISOURCE	LDA ==3.2E4	Divide 3.2E4 by the appropriate
4260	ISOURCE	LDB =Xvar	power of two, using BCD math.
4270	ISOURCE	XFR 4	
4280	ISOURCE	STB Oper_1	
4290	ISOURCE	LDA =Yvar	
4300	ISOURCE	STA Oper_2	
4310	ISOURCE	LDA Count	Decide whether to put the result
4320	ISOURCE	LDB =4	in R1, R2, or R3, depending on
4330	ISOURCE	MPY	Count.
4340	ISOURCE	ADA =R1	
4350	ISOURCE	STA Result	
4360	ISOURCE	LDA =2	
4370	ISOURCE	LDB =147155B	Now, finally, call the utility to
4380	ISOURCE	JSM Rel_math	perform the division.
4390	ISOURCE	ISZ Count	
4400	ISOURCE	LDA =3	Increment and check Count so as
4410	ISOURCE	CPA Count	to follow the loop three times.
4420	ISOURCE	JMP ++2	
4430	ISOURCE	JMP Get_freq	
4440	ISOURCE		
4450	ISOURCE	LDA =Array1	Zero out the entire count-to-
4460	ISOURCE	LDB =768	velocity conversion table so that
4470	ISOURCE	CLR 16	it must be recalculated for each
4480	ISOURCE	ADA =16	point. (This must be done if the
4490	ISOURCE	DSZ B	mixing frequencies or ranges are
4500	ISOURCE	JMP Continue	changed between counts.)
4510	ISOURCE		
4520	ISOURCE	LDA =Su	Set initial values of Su, Sv,
4530	ISOURCE	LDB =9	Sw, Suu, etc. to zero.
4540	ISOURCE	CLR 4	
4550	ISOURCE	ADA =4	
4560	ISOURCE	DSZ B	
4570	ISOURCE	JMP Clear	
4580	ISOURCE		
4590	ISOURCE	LDA =Theta	Convert Theta from degrees to
4600	ISOURCE	STA Oper_1	radians using the Rel_math
4610	ISOURCE	LDA =Rad	utility.
4620	ISOURCE	STA Oper_2	
4630	ISOURCE	LDA =Xvar	
4640	ISOURCE	STA Result	
4650	ISOURCE	LDA =2	
4660	ISOURCE	LDB =147155B	
4670	ISOURCE	JSM Rel_math	
4680	ISOURCE		
4690	ISOURCE	LDA =Xvar	Find the sine and cosine of
4700	ISOURCE	STA Oper_1	Theta, and store them in the
4710	ISOURCE	LDA =Sin	locations Sin and Cos, respectively.
4720	ISOURCE	STA Result	
4730	ISOURCE	LDA =1	
4740	ISOURCE	LDB =34213B	
4750	ISOURCE	JSM Rel_math	
4760	ISOURCE	LDA =Cos	
4770	ISOURCE	STA Result	
4780	ISOURCE	LDA =1	
4790	ISOURCE	LDB =34224B	
4800	ISOURCE	JSM Rel_math	

```

4810 ISOURCE |
4820 ISOURCE | The loop defined by Get_int calculates intermediate values
4830 ISOURCE | used in converting counts to velocities. These values are the
4840 ISOURCE | same for all samples in a point, so they can be calculated
4850 ISOURCE | separately. The loop calculates (Nub-NumixN) and (Mill*DfN),
4860 ISOURCE | where N is 1, 2, and 3.
4870 ISOURCE |
4880 ISOURCE | LDA =3
4890 ISOURCE | STA Count
4900 ISOURCE | Get_int: LDA Count
4910 ISOURCE | ADA =-1
4920 ISOURCE | SAL 2
4930 ISOURCE | STA Offset
4940 ISOURCE | LDA =Nub | Find NumixN=Nub-NumixN.
4950 ISOURCE | STA Oper_1
4960 ISOURCE | LDA =Numix1
4970 ISOURCE | ADA Offset
4980 ISOURCE | STA Oper_2
4990 ISOURCE | STA Result
5000 ISOURCE | LDA =2
5010 ISOURCE | LDB =146717B
5020 ISOURCE | JSM Rel_math
5030 ISOURCE | LDA =Df1 | Find DfN=Mill*DfN.
5040 ISOURCE | ADA Offset
5050 ISOURCE | STA Oper_1
5060 ISOURCE | STA Result
5070 ISOURCE | LDA =Mill
5080 ISOURCE | STA Oper_2
5090 ISOURCE | LDA =2
5100 ISOURCE | LDB =147037B
5110 ISOURCE | JSM Rel_math
5120 ISOURCE | DSZ Count
5130 ISOURCE | JMP Get_int
5140 ISOURCE |
5150 ISOURCE | The loop Begin is performed three times, once for U, V,
5160 ISOURCE | and W. Each time through the data array is read, a count (a
5170 ISOURCE | raw datum) is taken from it, and is converted into a velocity.
5180 ISOURCE | The velocity is stored in U, V, or W depending on whether
5190 ISOURCE | this is the first, second, or third iteration of the loop. The
5200 ISOURCE | first time any particular count is encountered, the velocity
5210 ISOURCE | corresponding to it is calculated using the intermediate values
5220 ISOURCE | found in Get_int, and the velocity is stored in a table. If that
5230 ISOURCE | count is found again in the data array, the corresponding vel-
5240 ISOURCE | ocity is looked up rather than being calculated again.
5250 ISOURCE |
5260 ISOURCE | LDA =0
5270 ISOURCE | STA Count
5280 ISOURCE | LDA =5
5290 ISOURCE | STA Check
5300 ISOURCE | LDA Ns
5310 ISOURCE | STA I
5320 ISOURCE |
5330 ISOURCE | Begin: LDA Count
5340 ISOURCE | STA B
5350 ISOURCE | SBL 2
5360 ISOURCE | STB Offset
5370 ISOURCE | LDB Ns | Figure out which element of the
5380 ISOURCE | MPY | data array we want to pick up.
5390 ISOURCE | ADA I
5400 ISOURCE | ADA =-1

```

5410	ISOURCE	STA Elementd	
5420	ISOURCE	LDA =Int	! Get a raw datum from the data array.
5430	ISOURCE	LDB =Arrayd	
5440	ISOURCE	JSM Get_element	
5450	ISOURCE	LDA Int	
5460	ISOURCE	LDB =1023	! Strip off the first six
5470	ISOURCE	AND B	! bits of the raw data word.
5480	ISOURCE	LDB Count	! See if Count = Check.
5490	ISOURCE	TCB	
5500	ISOURCE	ADB Check	
5510	ISOURCE	SZB Straight	! If true, use the stripped data
5520	ISOURCE	TCA	! word as an index. If not, use
5530	ISOURCE	ADA =1024	! 1024 minus the data word.
5540	ISOURCE	Straight: STA Int	! Store the count we have gotten.
5550	ISOURCE	LDA Count	! Now, use Count to find out
5560	ISOURCE	LDB =4096	! which lookup table array
5570	ISOURCE	MPY	! we want to use, and use the
5580	ISOURCE	LDB Int	! count we got from the data array
5590	ISOURCE	ADB =-1	! to find exactly where in the table
5600	ISOURCE	SBL 2	! we want to go.
5610	ISOURCE	ADA B	
5620	ISOURCE	ADA =1	
5630	ISOURCE	ADA =Array1	
5640	ISOURCE	STA Address	
5650	ISOURCE	LDA Address,1	! If that table entry is zero,
5660	ISOURCE	SZA Calculate	! calculate a velocity for it.
5670	ISOURCE	JMP Over	
5680	ISOURCE	Calculate: LDA =Int	
5690	ISOURCE	STA Oper_1	! Convert the count to a real number.
5700	ISOURCE	LDA =Yvar	
5710	ISOURCE	STA Result	
5720	ISOURCE	JSM Int_to_rel	
5730	ISOURCE	STB Oper_2	
5740	ISOURCE	LDA =R1	! Divide the range we found
5750	ISOURCE	ADA Offset	! earlier by the count to get a
5760	ISOURCE	STA Oper_1	! frequency.
5770	ISOURCE	LDA =Xvar	
5780	ISOURCE	STA Result	
5790	ISOURCE	LDA =2	
5800	ISOURCE	LDB =147155B	
5810	ISOURCE	JSM Rel_math	
5820	ISOURCE		
5830	ISOURCE	LDA =Num1x1	! Find (Nub-Num1xN)-FrequencyN.
5840	ISOURCE	ADA Offset	
5850	ISOURCE	STA Oper_1	
5860	ISOURCE	LDA =Xvar	
5870	ISOURCE	STA Oper_2	
5880	ISOURCE	LDA =Yvar	
5890	ISOURCE	STA Result	
5900	ISOURCE	LDA =2	
5910	ISOURCE	LDB =146717B	
5920	ISOURCE	JSM Rel_math	
5930	ISOURCE		
5940	ISOURCE	LDB Count	! If we are calculating U, reverse
5950	ISOURCE	ADB =-1	! the sign of ((Nub-Num1xN)-FrequencyN)
5960	ISOURCE	SBM **2	! so as to reverse the sign of U.
5970	ISOURCE	JMP Same sign	! Leave V and W alone.
5980	ISOURCE	LDA =Zero	
5990	ISOURCE	STA Oper_1	
6000	ISOURCE	LDA Result	


```

6010 ISOURCE          STA Oper_2
6020 ISOURCE          LDA =Xvar
6030 ISOURCE          STA Result
6040 ISOURCE          LDA =2
6050 ISOURCE          LDB =146717B
6060 ISOURCE          JSM Rel_math
6070 ISOURCE          Samesign: NOP
6080 ISOURCE          !
6090 ISOURCE          LDA Result          ! Find Velocity=((Nub-NumixN)
6100 ISOURCE          STA Oper_1          ! -FrequencyN)*(Mill*DfN)
6110 ISOURCE          LDA =Df1          ! and store in a place in the
6120 ISOURCE          ADA Offset          ! lookup table corresponding to
6130 ISOURCE          STA Oper_2          ! the data count.
6140 ISOURCE          LDA Address
6150 ISOURCE          ADA =-1
6160 ISOURCE          STA Result
6170 ISOURCE          LDA =2
6180 ISOURCE          LDB =147037B
6190 ISOURCE          JSM Rel_math
6200 ISOURCE          !
6210 ISOURCE          Over: LDA Address          ! Now transfer the velocity from
6220 ISOURCE          ADA =-1          ! the lookup table to U, V, or W,
6230 ISOURCE          LDB =U          ! as appropriate.
6240 ISOURCE          ADA Offset
6250 ISOURCE          XFR 4
6260 ISOURCE          !
6270 ISOURCE          ISZ Count
6280 ISOURCE          LDA =-3          ! Have U,V, and W all
6290 ISOURCE          ADA Count          ! been calculated?
6300 ISOURCE          Pause: NOP
6310 ISOURCE          SZA ++2          ! If not, go back again.
6320 ISOURCE          JMP Begin
6330 ISOURCE          STA Count          ! If so, set Count = 0.
6340 ISOURCE          !
6350 ISOURCE          ! Now we convert the w we have obtained (which is measured at
6360 ISOURCE          ! an angle Theta to the V-axis) to the W we want (which should
6370 ISOURCE          ! be measured at an angle of 90 degrees to the V-axis).
6380 ISOURCE          ! Thus find W=(Wv-V*Cos(Theta))/Sin(Theta).
6390 ISOURCE          !
6400 ISOURCE          LDA =V
6410 ISOURCE          STA Oper_1
6420 ISOURCE          LDA =Cos
6430 ISOURCE          STA Oper_2
6440 ISOURCE          LDA =Xvar
6450 ISOURCE          STA Result
6460 ISOURCE          LDA =2
6470 ISOURCE          LDB =147037B
6480 ISOURCE          JSM Rel_math
6490 ISOURCE          LDA Result
6500 ISOURCE          STA Oper_2
6510 ISOURCE          LDA =W
6520 ISOURCE          STA Oper_1
6530 ISOURCE          LDA =Yvar
6540 ISOURCE          STA Result
6550 ISOURCE          LDA =2
6560 ISOURCE          LDB =146717B
6570 ISOURCE          JSM Rel_math
6580 ISOURCE          LDA Result
6590 ISOURCE          STA Oper_1
6600 ISOURCE          LDA =Sin

```

```

6610 ISOURCE          STA Oper_2
6620 ISOURCE          LDA =W
6630 ISOURCE          STA Result
6640 ISOURCE          LDA =2
6650 ISOURCE          LDB =147155B
6660 ISOURCE          JSM Rel_math
6670 ISOURCE !
6680 ISOURCE ! Now take running sums of U, V, W, and several products
6690 ISOURCE ! of these velocities. The sums are taken using the utility
6700 ISOURCE ! "Add". The sums are calculated in an unusual sequence in
6710 ISOURCE ! order to reduce the number of steps needed to calculate
6720 ISOURCE ! them. Program steps which produce sums not considered
6730 ISOURCE ! necessary have been included, but these steps are preceded
6740 ISOURCE ! by the "!" comment marker.
6750 ISOURCE !
6760 ISOURCE          LDA =Su          ! Find Su=Su+U
6770 ISOURCE          LDB =U
6780 ISOURCE          JSM Add
6790 ISOURCE !
6800 ISOURCE          LDP Oper_2          ! Find U*U
6810 ISOURCE          STA Oper_1
6820 ISOURCE          LDA =Uu
6830 ISOURCE          STA Result
6840 ISOURCE          LDA =2
6850 ISOURCE          LDB =147037B
6860 ISOURCE          JSM Rel_math
6870 ISOURCE !
6880 ISOURCE          LDA =Suu          ! Find Su=Su+(U*U)
6890 ISOURCE          LDB Result
6900 ISOURCE          JSM Add
6910 ISOURCE !
6920 ISOURCE          LDA =Sv          ! Find Su=Sv+V
6930 ISOURCE          LDB =V
6940 ISOURCE          JSM Add
6950 ISOURCE !
6960 ISOURCE          LDA =U          ! Find U*V
6970 ISOURCE          STA Oper_1
6980 ISOURCE          LDA =Uv
6990 ISOURCE          STA Result
7000 ISOURCE          LDA =2
7010 ISOURCE          LDB =147037B
7020 ISOURCE          JSM Rel_math
7030 ISOURCE !
7040 ISOURCE          LDA =Suv          ! Find Suv=Suv+(U*V)
7050 ISOURCE          LDB Result
7060 ISOURCE          JSM Add
7070 ISOURCE !
7080 ISOURCE          LDA =V          ! Find V*V
7090 ISOURCE          STA Oper_1
7100 ISOURCE          STA Oper_2
7110 ISOURCE          LDA =Xvar
7120 ISOURCE          STA Result
7130 ISOURCE          LDA =2
7140 ISOURCE          LDB =147037B
7150 ISOURCE          JSM Rel_math
7160 ISOURCE !
7170 ISOURCE          LDA =Svv          ! Find Svv=Svv+(V*V)
7180 ISOURCE          LDB Result
7190 ISOURCE          JSM Add
7200 ISOURCE !

```

```

7210 ISOURCE      LDA =Sw          ' Find Sw=Sw+W
7220 ISOURCE      LDB =W
7230 ISOURCE      JSM Add
7240 ISOURCE
7250 ISOURCE      LDA =U          ' Find U*W
7260 ISOURCE      STA Oper_1
7270 ISOURCE      LDA =Xvar
7280 ISOURCE      STA Result
7290 ISOURCE      LDA =2
7300 ISOURCE      LDB =147037B
7310 ISOURCE      JSM Rel_math
7320 ISOURCE
7330 ISOURCE      LDA =Suw        ' Find Suw=Suw+(U*W)
7340 ISOURCE      LDB Result
7350 ISOURCE      JSM Add
7360 ISOURCE
7370 ISOURCE      LDA =W          ' Find V*W
7380 ISOURCE      STA Oper_1
7390 ISOURCE      LDA =V
7400 ISOURCE      STA Oper_2
7410 ISOURCE      LDA =Xvar
7420 ISOURCE      STA Result
7430 ISOURCE      LDA =2
7440 ISOURCE      LDB =147037B
7450 ISOURCE      JSM Rel_math
7460 ISOURCE
7470 ISOURCE      LDA =Suw        ' Find Suw=Suw+(V*W)
7480 ISOURCE      LDB Result
7490 ISOURCE      JSM Add
7500 ISOURCE
7510 ISOURCE      LDA =W          ' Find W*W
7520 ISOURCE      STA Oper_1
7530 ISOURCE      STA Oper_2
7540 ISOURCE      LDA =Xvar
7550 ISOURCE      STA Result
7560 ISOURCE      LDA =2
7570 ISOURCE      LDB =147037B
7580 ISOURCE      JSM Rel_math
7590 ISOURCE
7600 ISOURCE      LDA =Suw        ' Find Suw=Suw+(W*W)
7610 ISOURCE      LDB Result
7620 ISOURCE      JSM Add
7630 ISOURCE
7640 ISOURCE      DSZ I            Cont nue to calculate running
7650 ISOURCE      JMP Begin        ' sums until out of samples.
7660 ISOURCE
7670 ISOURCE      ' Place the finished sums in the COMMON region so that
7680 ISOURCE      ' the BASIC program has access to them, and then return to
7690 ISOURCE      ' the BASIC program.
7700 ISOURCE
7710 ISOURCE      LDA =Su
7720 ISOURCE      LDB =Su_par
7730 ISOURCE      JSM Put_value
7740 ISOURCE      LDA =Su
7750 ISOURCE      LDB =Su_par
7760 ISOURCE      JSM Put_value
7770 ISOURCE      LDA =Sw
7780 ISOURCE      LDB =Sw_par
7790 ISOURCE      JSM Put_value
7800 ISOURCE      LDA =Suw

```

```

7810 ISOURCE      LDB =Suu_par
7820 ISOURCE      JSM Put_value
7830 ISOURCE      LDA =Suu
7840 ISOURCE      LDB =Suu_par
7850 ISOURCE      JSM Put_value
7860 ISOURCE      LDA =Suu
7870 ISOURCE      LDB =Suu_par
7880 ISOURCE      JSM Put_value
7890 ISOURCE      LDA =Suu
7900 ISOURCE      LDB =Suu_par
7910 ISOURCE      JSM Put_value
7920 ISOURCE      LDA =Suu
7930 ISOURCE      LDB =Suu_par
7940 ISOURCE      JSM Put_value
7950 ISOURCE      LDA =Suu
7960 ISOURCE      LDB =Suu_par
7970 ISOURCE      JSM Put_value
7980 ISOURCE      RET 1
7990 ISOURCE      |
8000 ISOURCE      | The utility "Add" is used to add up the running sums.
8010 ISOURCE      |
8020 ISOURCE      Add:      ISZ Utilcount
8030 ISOURCE      STA Oper_1
8040 ISOURCE      STB Oper_2
8050 ISOURCE      STA Result
8060 ISOURCE      LDA =2
8070 ISOURCE      LDB =146721B
8080 ISOURCE      JSM Rel_math
8090 ISOURCE      DSZ Utilcount
8100 ISOURCE      RET 1
8110 ISOURCE      JSM Utlend
8120 ISOURCE      |
8130 ISOURCE      LIT 200
8140 ISOURCE      END Find_val
8150 ISOURCE      |
8160 ISOURCE      NAM Data_trans
8170 ISOURCE      |
8180 ISOURCE      | Subroutine to transfer raw data from input array
8190 ISOURCE      | to storage array.
8200 ISOURCE      |
8210 ISOURCE      EXT Get_info      | Declare subroutines stored
8220 ISOURCE      EXT Get_value     | outside of the main program.
8230 ISOURCE      EXT Get_element
8240 ISOURCE      EXT Put_element
8250 ISOURCE      |
8260 ISOURCE      COM
8270 ISOURCE      Data_par: INT (*)      | Declare common variables.
8280 ISOURCE      D1_par:  INT (*)
8290 ISOURCE      Ns_par:   INT
8300 ISOURCE      |
8310 ISOURCE      Arrayd:   BSS 39
8320 ISOURCE      Elementd: EQU Arrayd+16
8330 ISOURCE      Array1:   BSS 39
8340 ISOURCE      Element1: EQU Array1+16
8350 ISOURCE      Ns:       BSS 1
8360 ISOURCE      Count1:   BSS 1
8370 ISOURCE      Count2:   BSS 1
8380 ISOURCE      I:        BSS 1
8390 ISOURCE      Int:      BSS 1
8400 ISOURCE      |

```

```

8410 ISOURCE SUB
8420 ISOURCE Data_trans: LDA =Arrayd      ! Get parameters of data storage array.
8430 ISOURCE LDB =Data_par
8440 ISOURCE JSM Get_info
8450 ISOURCE LDA =Array1      ! Get parameters of data input array.
8460 ISOURCE LDB =D1_par
8470 ISOURCE JSM Get_info
8480 ISOURCE LDA =Ns          ! Get number of samples.
8490 ISOURCE LDB =Ns_par
8500 ISOURCE JSM Get_value
8510 ISOURCE !
8520 ISOURCE LDA =0
8530 ISOURCE STA I
8540 ISOURCE Start1: LDA =0
8550 ISOURCE STA Count1
8560 ISOURCE LDA =2
8570 ISOURCE STA Count2
8580 ISOURCE !
8590 ISOURCE Start2: LDA I
8600 ISOURCE LDB =6
8610 ISOURCE MPY
8620 ISOURCE ADA Count2
8630 ISOURCE STA Element1
8640 ISOURCE LDA =Int
8650 ISOURCE LDB =Array1
8660 ISOURCE JSM Get_element
8670 ISOURCE !
8680 ISOURCE LDA Count1
8690 ISOURCE LDB Ns
8700 ISOURCE MPY
8710 ISOURCE ADA I
8720 ISOURCE STA Elementd
8730 ISOURCE LDA =Int
8740 ISOURCE LDB =Arrayd
8750 ISOURCE JSM Put_element
8760 ISOURCE ISZ Count1
8770 ISOURCE ISZ Count2
8780 ISOURCE LDA =-3
8790 ISOURCE ADA Count1
8800 ISOURCE SZA **2
8810 ISOURCE JMP Start2
8820 ISOURCE !
8830 ISOURCE ISZ I
8840 ISOURCE LDA I
8850 ISOURCE TCA
8860 ISOURCE ADA Ns
8870 ISOURCE SZA **2
8880 ISOURCE JMP Start1
8890 ISOURCE RET 1
8900 ISOURCE END Data_trans
8910 !
8920 ! *****
8930 ! ** HISTOGRAM SUBPROGRAM
8940 ! *****
8950 !
8960 Hist:      ! Subroutine to produce online
8970             ! histograms of raw LDV data.
8980 ON KEY #0 GOTO 9270
8990 Nn=9
9000 Ns_temp=Ns

```

```

9010 |
9020 INPUT "No. of data samples per point (2000 maxm.) :",Ns
9030 |
9040 REDIM Data(3,Ns)
9050 REDIM D1(Ns,6)
9060 DISP "Press CONT to initiate data acquisition, press K0 to return to main
program."
9070 PAUSE
9080 GCLEAR
9090 GRAPHICS
9100 |
9110 RESET 10
9120 CONTROL MASK 10;1
9130 WRITE IO 10,5;0
9140 WRITE IO 10,5;1 | start handshake by setting CTL0
9150 Nt=6*Ns
9160 FOR I=1 TO 5
9170 Dummy=READBIN(10)
9180 NEXT I
9190 |
9200 ENTER 10 WFHS Nt NOFORMAT;D1(*) |fast data acquisition
9210 WRITE IO 10,5;0
9220 ICALL Data_trans
9230 ICALL Draw_hist
9240 IF Ns=2000 THEN GOTO 9270
9250 Nn=1
9260 GOTO 9110
9270 IF Ns=2000 THEN DUMP GRAPHICS
9280 WRITE IO 10,5;0
9290 Ns=Nn_temp
9300 REDIM Data(3,Ns)
9310 REDIM D1(Ns,6)
9320 EXIT GRAPHICS
9330 RETURN
9340 |
9350 ISOURCE NAM Draw_hist
9360 ISOURCE |
9370 ISOURCE | The subroutine works by first going to the samples-per-count
9380 ISOURCE | tables and using them to draw the old histograms in black
9390 ISOURCE | (Bit=0) to erase them. Then it calculates new samples-per-
9400 ISOURCE | count tables from the data acquired from the LDV, and uses
9410 ISOURCE | the new tables to draw histograms in white (Bit=1). Then the
9420 ISOURCE | subroutine returns to the main program. The very first time
9430 ISOURCE | (determined by Ntimes(=Nn)) the subroutine is called it doesn't
9440 ISOURCE | erase the old histograms because there aren't any.
9450 ISOURCE |
9460 ISOURCE EXT Get_value | Declare subroutines stored
9470 ISOURCE EXT Get_info | outside of the main program.
9480 ISOURCE EXT Get_element
9490 ISOURCE |
9500 ISOURCE COM
9510 ISOURCE Data_par: INT (*) | Declare common variables.
9520 ISOURCE D1_par: INT (*)
9530 ISOURCE Ns_par: INT
9540 ISOURCE Ntime_par: INT
9550 ISOURCE |
9560 ISOURCE Arrayd: BSS 39 | Reserve space for data array
9570 ISOURCE Elementd: EQU Arrayd+16 | descriptor.
9580 ISOURCE Array1: BSS 1024 | Reserve space for tables which hold
9590 ISOURCE Array2: BSS 1024 | the number of samples per count.
9600 ISOURCE Array3: BSS 1024

```

```

9610 ISOURCE Ns: BSS 1
9620 ISOURCE Count: BSS 1 | Count and I are general purpose index vari-
9630 ISOURCE I: BSS 1 | ables. Count is usually 0, 1, or 2, to denote
9640 ISOURCE Check: BSS 1 | whether U,V, or W is being calculated.
9650 ISOURCE Int: BSS 1 | Int, Address, and Offset are all general
9660 ISOURCE Address: BSS 1 | purpose storage areas.
9670 ISOURCE X_coord: BSS 1
9680 ISOURCE Y_coord: BSS 1
9690 ISOURCE Bit: BSS 1
9700 ISOURCE Ntimes: BSS 1
9710 ISOURCE |
9720 ISOURCE LIT 30
9730 ISOURCE |
9740 ISOURCE SUB
9750 ISOURCE Draw_hist: LDA =Ns | Get number of samples.
9760 ISOURCE LDB =Ns_par
9770 ISOURCE JSM Get_value
9780 ISOURCE LDA =Arrayd | Get parameters of data array.
9790 ISOURCE LDB =Data_par
9800 ISOURCE JSM Get_info
9810 ISOURCE LDA =Ntimes | Get Ntimes, which tells if this is
9820 ISOURCE LDB =Ntime_par | the first time Draw_hist is being
9830 ISOURCE JSM Get_value | called.
9840 ISOURCE |
9850 ISOURCE LDA =0 | If this is the first time Draw_hist
9860 ISOURCE STA Bit | is being called, then jump to the
9870 ISOURCE LDA Ntimes | data acquisition section. If not then
9880 ISOURCE RZA #+2 | write over the old histogram with
9890 ISOURCE JMP Acquire | black to erase it
9900 ISOURCE |
9910 ISOURCE Do_graph: LDA =0 | Produce histograms for all three
9920 ISOURCE STA Count | channels.
9930 ISOURCE Make_hist: LDA Count | Go to the end of the appropriate
9940 ISOURCE ADA =1 | samples-per-count table.
9950 ISOURCE LDB #1024
9960 ISOURCE STB I
9970 ISOURCE MPY
9980 ISOURCE ADA =Array1
9990 ISOURCE ADA =-1
10000 ISOURCE STA Address
10010 ISOURCE Make_row: LDB Address,I | Add together each adjacent pair of
10020 ISOURCE DSZ Address | entries in the table.
10030 ISOURCE ADB Address,I
10040 ISOURCE DSZ Address
10050 ISOURCE DSZ I
10060 ISOURCE RZB #+2 | If the result is zero, go to the next
10070 ISOURCE JMP Skip_bits | pair of entries. If not, make sure
10080 ISOURCE STB Int | the sum is <150 and then draw a
10090 ISOURCE ADB #-150 | column with height equal to the sum.
10100 ISOURCE SBM #+3
10110 ISOURCE LDB #150
10120 ISOURCE STB Int
10130 ISOURCE |
10140 ISOURCE LDA | | Calculate the X-coordinate of the
10150 ISOURCE SAR 1 | column.
10160 ISOURCE ADA #20
10170 ISOURCE STA X_coord
10180 ISOURCE |
10190 ISOURCE Make_col: LDA #150 | Calculate the Y-coordinate of the
10200 ISOURCE LDB Count | top of the column.

```

```

10210 ISOURCE      ADB =1
10220 ISOURCE      MPY
10230 ISOURCE      LDB Int
10240 ISOURCE      TCB
10250 ISOURCE      ADA B
10260 ISOURCE      STA Y_coord
10270 ISOURCE      |
10280 ISOURCE      LDA =13      | Prepare the graphics screen for
10290 ISOURCE      STA Pa      | input.
10300 ISOURCE      LDA =518
10310 ISOURCE      SFC *
10320 ISOURCE      STA R5
10330 ISOURCE      LDA Y_coord  | Calculate word location.
10340 ISOURCE      LDB =36
10350 ISOURCE      MPY
10360 ISOURCE      LDB X_coord
10370 ISOURCE      SBR 4
10380 ISOURCE      ADA B
10390 ISOURCE      CMA
10400 ISOURCE      SFC *
10410 ISOURCE      STA R4      | Input word location.
10420 ISOURCE      STA R7
10430 ISOURCE      |
10440 ISOURCE      LDA X_coord  | Calculate bit location.
10450 ISOURCE      AND =178
10460 ISOURCE      LDB Bit
10470 ISOURCE      SBL 15
10480 ISOURCE      IOR B
10490 ISOURCE      SFC *
10500 ISOURCE      STA R4      | Input bit location.
10510 ISOURCE      STA R7
10520 ISOURCE      DSZ Int
10530 ISOURCE      JMP Make_bit
10540 ISOURCE      |
10550 ISOURCE      Skip_bits: DSZ I
10560 ISOURCE      JMP Make_rnd
10570 ISOURCE      |
10580 ISOURCE      ISZ Count    | Make histograms for all three
10590 ISOURCE      LDA =-3      | channels.
10600 ISOURCE      ADA Count
10610 ISOURCE      SZA **2
10620 ISOURCE      JMP Make_hist
10630 ISOURCE      |
10640 ISOURCE      Acquire: LDA Bit    | If Bit=1 then the new histograms
10650 ISOURCE      SZA **2      | have been drawn. Go back to the
10660 ISOURCE      JMP Stop     | main program. If not, continue.
10670 ISOURCE      |
10680 ISOURCE      LDA =Array1    | Zero out the tables which hold the
10690 ISOURCE      LDB =192      | number of samples per count.
10700 ISOURCE      Clear:  CLR 16
10710 ISOURCE      ADA =16
10720 ISOURCE      DSZ B
10730 ISOURCE      JMP Clear
10740 ISOURCE      |
10750 ISOURCE      LDA =0        | Prepare to write new samples-per-
10760 ISOURCE      STA Count     | count tables.
10770 ISOURCE      LDA =5
10780 ISOURCE      STA Check
10790 ISOURCE      LDA Ns
10800 ISOURCE      STA I

```


10810	ISOURCE		LDA Count	
10820	ISOURCE	Begin:	LDB Ns	Figure out which element of the
10830	ISOURCE		MPY	data array we want to pick up.
10840	ISOURCE		ADA I	
10850	ISOURCE		ADA =-1	
10860	ISOURCE		STA Elementd	
10870	ISOURCE		LDA =Int	Get a raw datum from the data array.
10880	ISOURCE		LDB =Arrayd	
10890	ISOURCE		JSM Get_element	
10900	ISOURCE		LDA Int	
10910	ISOURCE		LDB =1023	Strip off the first six
10920	ISOURCE		AND B	bits of the raw data word.
10930	ISOURCE		LDB Count	See if Count = Check.
10940	ISOURCE		TCB	
10950	ISOURCE		ADB Check	
10960	ISOURCE		SZB Straight	If true, use the modified data
10970	ISOURCE		TCA	word as an index. If not, use
10980	ISOURCE		ADA =1324	1024 minus the data word.
10990	ISOURCE		STA Int	Store the count we have gotten.
11000	ISOURCE	Straight:	LDA =1024	Increment the appropriate place
11010	ISOURCE		LDB Count	in the samples-per-count table
11020	ISOURCE		MPY	by one.
11030	ISOURCE		ADA =Array1	
11040	ISOURCE		ADA Int	
11050	ISOURCE		ISZ A,I	
11060	ISOURCE			
11070	ISOURCE		ISZ Count	
11080	ISOURCE		LDA =-3	Have U,V, and W all been done?
11090	ISOURCE		ADA Count	
11100	ISOURCE		SZA ++2	If not, go back again.
11110	ISOURCE		JMP Begin	
11120	ISOURCE		STA Count	If so, set Count=0.
11130	ISOURCE			
11140	ISOURCE		DSZ I	Continue to fill samples-per-count
11150	ISOURCE		JMP Begin	tables until out of samples.
11160	ISOURCE			
11170	ISOURCE		ISZ Bit	Now set Bit=1 and go back and draw
11180	ISOURCE		JMP Do_graph	the new histograms in white.
11190	ISOURCE	Stop:	RET 1	
11200	ISOURCE			
11210	ISOURCE		END Draw_hist	
11220	ISOURCE			

```

10  REM PROGRAM STAT
20  | PROGRAM TO REDUCE RAW DATA FROM THREE-COMPONENT LDV SYSTEM
30  | Input is 3 channels of raw LV data and calibration constants
40  | Output is the three components of mean velocity, all the
50  | components of Reynolds stress and some selected third-order
60  | products. Can print out counts which are excessively far
70  | from the mean. Can display histograms of raw and
80  | filtered data for all 3 channels. Plots data at end and displays
90  | reduced data for entire run. Can write a summary file containing
100 | all reduced data and calibration constants.
110 OPTION BASE 1
120 ICOM 10000
130 IDELETE ALL
140 IASSEMBLE Find_vel
150 COM INTEGER Data(3,2000),Ns,Ngs,Countbin(3,1024)
160 COM INTEGER Range1,Range2,Range3
170 COM REAL Sdev,Df1,Df2,Df3
180 COM REAL Theta,Nub,Numix1,Numix2,Numix3
190 COM REAL Su,Sv,Sw,Suu,Suv,Suw,Suv,Suw,Suv
200 COM REAL Suuv,Suuv,Suuv,Suuv,Suuv
210 |
220 DIM L1(3),R1(3),L11(3),U11(3),Scale(3)
230 DIM Date$(80),File$(6),Name$(4),Titl$(80)
240 |
250 REAL Xpos,Ypos,Zpos
260 REAL Point(20,15)
270 REAL Re,Ue
280 |
290 INTEGER Filno,A,Run
300 |
310 DATA 0,0,0,0
320 READ Max_y,Min_v,Filter,Sdev
330 DATA 75.64,45.64,15.64,99.,69.,39.
340 MAT READ L11,U11
350 DATA 5,46,87,41,82,123
360 MAT READ L1,R1
370 DATA -.8,2.4,0.,.035,-.0025,.01
380 READ Llim(1),Ulim(1),Llim(2),Ulim(2),Llim(3),Ulim(3)
390 DATA .4,.005,.0025
400 MAT READ Scale
410 |
420 PRINT
430 PRINT " ** << PROGRAM STAT : 3-COMPONENT VELOCITY DATA >> ** "
440 PRINT
450 PRINT "PROGRAM STRUCTURE"
460 PRINT "1. Read raw data from floppy disc"
470 PRINT "2. Convert to velocity"
480 PRINT "3. Calculate statistics"
490 PRINT "4. Print results"
500 PRINT "5. Write to disc file"
510 PRINT
520 |
530 | ** Read raw data from floppy disc **
540 |
550 INPUT "Enter parent filename (or E to exit program) :",Name$
560 IF Name$="E" THEN GOTO 1780
570 INPUT "Enter no. of first and last data files ",File1,Nf
580 Kill w=3
590 INPUT "Is this a two-channel or three-channel run (2/3, Default 3) ?",Kill
600 Hist$="Y"

```

```

610 INPUT "Do you wish to see histograms of the data points (Y/N, Default Y)
",Hist$
620 Filters$="N"
630 INPUT "Do you wish to filter the output (Y/N, Default N) ?",Filters$
640 IF Filters$="N" THEN GOTO 660
650 INPUT "Enter standard deviation to throw out",Sdev
660 DISP "READING RAW DATA FROM DISK"
670 MASS STORAGE IS ":H8,0,0"
680 PRINTER IS 0
690 Filno=File1
700 IF Filno>Nf THEN GOTO 1740
710 Files$=Name$&VAL$(Filno)
720 ASSIGN Files$ TO #1
730 I
740 READ #1;Dates$
750 READ #1;Tit1$
760 READ #1;Name$
770 READ #1;Dn
780 READ #1;Nub,Num1x1,Num1x2,Num1x3,Theta,Run
790 READ #1;Vref,Ue,Df1,Df2,Df3
800 READ #1;Xpos,Ypos,Zpos,Ns
810 MAT READ #1;Data
820 I
830 PRINT Tit1$
840 PRINT "DATE AND TIME OF TEST :",Dates$
850 PRINT "FILE NAME ON DISK :",Name$
860 PRINT "DISK NUMBER :",Dn
870 PRINT "RUN NO. (POINT NO. IN PROFILE) :",Run
880 PRINT "X,Y,Z = ",Xpos,Ypos,Zpos
890 I
900 I CALCULATE VELOCITIES
910 DISP "CALCULATING VELOCITIES AND TAKING RUNNING SUMS"
920 I
930 ICALL Find_vel
940 I
950 I CALCULATE STATISTICS
960 DISP "CALCULATING STATISTICS"
970 I
980 IF Kill_w=3 THEN GOTO 1030
990 DATA 0,0,0,0,0,0,0
1000 RESTORE 990
1010 READ Sw,Sww,Suw,Svw,Suww,Suww,Suww
1020 I
1030 Ubar=Su/Ngs
1040 Vbar=Su/Ngs
1050 Wbar=Sw/Ngs
1060 Upr12=Suw/Ngs-Ubar*Ubar
1070 Vpr12=Swv/Ngs-Vbar*Vbar
1080 Wpr12=Sww/Ngs-Wbar*Wbar
1090 Uvbar=Suw/Ngs-Ubar*Vbar
1100 Uwbar=Suw/Ngs-Ubar*Wbar
1110 Vwbar=Swv/Ngs-Vbar*Wbar
1120 Uuvbar=Suww/Ngs-2*Ubar*Suw/Ngs-Vbar*Suw/Ngs+2*Vbar*Ubar*Ubar
1130 Uvvbar=Suww/Ngs-2*Vbar*Suw/Ngs-Ubar*Suw/Ngs+2*Ubar*Vbar*Vbar
1140 Uuwbar=Suww/Ngs-2*Ubar*Suw/Ngs-Wbar*Suw/Ngs+2*Wbar*Ubar*Ubar
1150 Uwwbar=Suww/Ngs-2*Wbar*Suw/Ngs-Ubar*Suw/Ngs+2*Ubar*Wbar*Wbar
1160 Uvwbar=Suww/Ngs-Wbar*Suw/Ngs-Ubar*Suw/Ngs-Vbar*Suw/Ngs+2*Ubar*Vbar*Wbar
1170 I
1180 I STORE RESULTS IN ARRAY FOR PLOTTING
1190 I
1200 F1=Filno-File1+1

```

```

1210 Point(F1,1)=Ypos
1220 Point(F1,2)=Ubar
1230 Point(F1,3)=Vbar
1240 Point(F1,4)=Wbar
1250 Point(F1,5)=Upri2
1260 Point(F1,6)=Vpri2
1270 Point(F1,7)=Wpri2
1280 Point(F1,8)=Uvbar
1290 Point(F1,9)=Uwbar
1300 Point(F1,10)=Vwbar
1310 Point(F1,11)=Uuvbar
1320 Point(F1,12)=Uvvbar
1330 Point(F1,13)=Uuwbar
1340 Point(F1,14)=Uwwbar
1350 Point(F1,15)=Uvwbar
1360 IF Ypos<Max_y THEN GOTO 1390
1370 Max_y=Ypos
1380 Min_u=Ubar
1390 IF Ypos>Min_y THEN GOTO 1450
1400 Min_y=Ypos
1410 Max_u=Ubar
1420 !
1430 ! PRINT RESULTS
1440 !
1450 PRINT
1460 PRINT "Bragg shift frequency = ",Nub
1470 PRINT "Mixing frequencies=",Numix1,Numix2,Numix3
1480 PRINT "Third beam angle = ",Theta
1490 PRINT "Run number = ",Run
1500 PRINT "Tunnel reference voltage (volts) = ",Vref
1510 PRINT "Fringe spacings = ",Df1,Df2,Df3
1520 PRINT "No. of samples = ",Ns
1530 PRINT "Ranges = ",Range1,Range2,Range3
1540 PRINT "Filter [no. of std. deviations] = ",Sdev
1550 PRINT "Adjusted no. of samples =",Ngs
1560 PRINT "Ubar = ",PROUND(Ubar,-4)
1570 PRINT "Vbar = ",PROUND(Vbar,-4)
1580 PRINT "Wbar = ",PROUND(Wbar,-4)
1590 PRINT "Upri2 = ",PROUND(Upri2,-4)
1600 PRINT "Vpri2 = ",PROUND(Vpri2,-4)
1610 PRINT "Wpri2 = ",PROUND(Wpri2,-4)
1620 PRINT "Uvbar = ",PROUND(Uvbar,-4)
1630 PRINT "Uwbar = ",PROUND(Uwbar,-4)
1640 PRINT "VWbar = ",PROUND(Vwbar,-4)
1650 PRINT "UUVbar = ",PROUND(Uuvbar,-5)
1660 PRINT "UVVbar = ",PROUND(Uvvbar,-5)
1670 PRINT "Uuwbar = ",PROUND(Uuwbar,-5)
1680 PRINT "Uwwbar = ",PROUND(Uwwbar,-5)
1690 PRINT "Uvwbar = ",PROUND(Uvwbar,-5)
1700 PRINT
1710 IF Hist$="Y" THEN GOSUB Histogram
1720 Filno=Filno+1
1730 GOTO 700
1740 Ue=Max_u-Min_u
1750 Plotr$="Y"
1760 INPUT " Do you wish to see the turbulence quantities (Y/N, Default Y) ? ",Plotr$
1770 IF Plotr$="Y" THEN GOSUB Plot
1780 MASS STORAGE IS ":H8,0,1"
1790 PRINTER IS 16
1800 END

```

```

1810 |
1820 | *****PLOT DATA*****
1830 |
1840 Plot: PRINTER IS 16
1850 PRINT "Estimate of normalizing velocity is :";Ue
1860 INPUT "Enter correct normalizing velocity if different.",Ue
1870 Graph$="Y"
1880 INPUT "Do you wish to graph the turbulence quantities (Y/N, Default Y) ?",
Graph$
1890 PRINTER IS 0
1900 PRINT "Normalizing velocity is :";Ue
1910 |
1920 FOR I=1 TO 4
1930 DATA 1,2,2,3
1940 READ Exp
1950 Kn=3
1960 IF I=4 THEN Kn=5
1970 FOR J=1 TO Nf-Nfile+1
1980 FOR K=1 TO Kn
1990 Index=(I-1)*3+K+1
2000 Point(J,Index)=Point(J,Index)/Ue^Exp
2010 IF Index>10 THEN Point(J,Index)=Point(J,Index)*1000
2020 NEXT K
2030 NEXT J
2040 NEXT I
2050 IF Graph$="N" THEN GOTO 2060
2060 |
2070 PLOTTER IS 13,"GRAPHICS"
2080 GRAPHICS
2090 DEG
2100 |
2110 Ymax=Point(1,1)
2120 Ymin=Ymax
2130 FOR J=1 TO Nf-File1+1
2140 IF Point(J,1)>Ymax THEN Ymax=Point(J,1)
2150 IF Point(J,1)<Ymin THEN Ymin=Point(J,1)
2160 NEXT J
2170 |
2180 FOR I=1 TO 3
2190 LIMIT 0,184.47,0,149.2
2200 LOCATE L1(I),R1(I),39,99
2210 CLIP L1(I),R1(I),39,99
2220 |
2230 Xmax=Point(1,(I-1)*3+2)
2240 Xmin=Xmax
2250 FOR J=1 TO Nf-File1+1
2260 FOR K=1 TO 3
2270 IF Point(J,(I-1)*3+K+1)>Xmax THEN Xmax=Point(J,(I-1)*3+K+1)
2280 IF Point(J,(I-1)*3+K+1)<Xmin THEN Xmin=Point(J,(I-1)*3+K+1)
2290 NEXT K
2300 NEXT J
2310 |
2320 SCALE L1m(I),U1m(I),Ymin,Ymax
2330 FRAME
2340 AXES Scale(I),.1,0,0
2350 UNCLIP
2360 |
2370 LOG 0
2380 CSIZE 2.5
2390 Ypos=Ymin
2400 MOVE L1m(I)+Scale(I)/5,Ypos

```

```

2410 DRAW Llim(I),Ypos
2420 SETGU
2430 RPLLOT 2,0,-2
2440 SETUU
2450 LABEL DROUND(Ypos,2)
2460 Ypos=Ypos+.1
2470 IF Ypos<=Ymax THEN GOTO 2480
2480 !
2490 LONG 6
2500 Xpos=Llim(I)
2510 MOVE Xpos,Ymin-.04
2520 LABEL DROUND(Xpos,2)
2530 MOVE Xpos,Ymin
2540 DRAW Xpos,Ymin+.05
2550 Xpos=Xpos+Scale(I)
2560 IF Xpos<Ulim(I) THEN GOTO 2510
2570 !
2580 FOR J=1 TO Nf-File1+1
2590 Ypos=Point(J,1)
2600 FOR K=1 TO 3
2610 MOVE Point(J,(I-1)*3+K+1),Ypos
2620 SETGU
2630 ON K GOTO 2640,2700,2760
2640 FOR Arc=0 TO 360 STEP 20
2650 PDIR Arc
2660 RPLLOT .5,0
2670 NEXT Arc
2680 PDIR 0
2690 GOTO 2810
2700 RPLLOT .5,.5,-2
2710 RPLLOT .5,-.5,-1
2720 RPLLOT -.5,-.5,-1
2730 RPLLOT -.5,.5,-1
2740 RPLLOT .5,.5,-1
2750 GOTO 2810
2760 RPLLOT 0,.5,-2
2770 RPLLOT 0,-.5,-1
2780 RPLLOT .5,0,-2
2790 RPLLOT -.5,0,-1
2800 GOTO 2810
2810 SETUU
2820 NEXT K
2830 NEXT J
2840 NEXT I
2850 DUMP GRAPHICS
2860 IMAGE 2X"Y"10X, 'u2/Uo2",3X,"v2/Uo2",3X,"w2/Uo2",5X,"uv/Uo2",3X,"uw/Uo2",3X
,"vw/Uo2",/
2870 PRINT USING 2860
2880 FOR I=1 TO Nf-File1+1
2890 IMAGE MDD.0D,6X,MD.DDDD,2X,MD.DDDD,2X,MD.DDDD,4X,MD.DDDD,2X,MD.DDDD,2X,MD.
DDDD
2900 PRINT USING 2890;Point(I,1),Point(I,5),Point(I,6),Point(I,7),Point(I,8),Po
int(I,9),Point(I,10)
2910 NEXT I
2920 IMAGE 2/,3X,"Y", 9X,"uuu/Uo2",3X,"uuu/Uo3",5X,"uuw/Uo3",3X,"uuw/Uo3",5X,"u
vw/Uo3",/
2930 PRINT USING 2920
2940 FOR I=1 TO Nf-File1+1
2950 IMAGE MDD.0D,6X,MD.5D,2X,MD.5D,4X,MD.5D,2X,MD.5D,4X,MD.5D
2960 PRINT USING 2950;Point(I,1),Point(I,11),Point(I,12),Point(I,13),Point(I,14
),Point(I,15)

```

```

2970 NEXT I
2980 !
2990 ! WRITE SUMMARY DATA FILE
3000 !
3010 Sum$="N"
3020 INPUT "Do you wish to write a Summary Data File (Y/N Default N) ? ",Sum$
3030 IF Sum$="N" THEN GOTO 3210
3040 File$=Name$
3050 DISP "File ";File$;" being written to disk"
3060 MASS STORAGE IS ":H8,0,0"
3070 CREATE File$,40
3080 ASSIGN File$ TO #1
3090 PRINT #1;Date$
3100 PRINT #1;Titl$
3110 PRINT #1;Name$
3120 PRINT #1;Dn
3130 PRINT #1;Nub,Num1x1,Num1x2,Num1x3,Theta,Run
3140 PRINT #1;Vref,Ux,Df1,Df2,Df3
3150 PRINT #1;Ns,File1,Nf
3160 FOR I=1 TO Nf-Nfile+1
3170 FOR J=1 TO 15
3180 PRINT #1;Point(I,J)
3190 NEXT J
3200 NEXT I
3210 RETURN
3220 !
3230 ! *****DRAW HISTOGRAMS*****
3240 !
3250 Histogram: PLOTTER IS 13,"GRAPHICS"
3260 GRAPHICS
3270 K=1
3280 FOR I=1 TO 3
3290 Cmax=0
3300 FOR J=1 TO 1023 STEP 2
3310 Countbin(I,J)=Countbin(I,J)+Countbin(I,J+1)
3320 IF Countbin(I,J)>Cmax THEN Cmax=Countbin(I,J)
3330 NEXT J
3340 IF (Cmax=0) OR (Cmax=2000) THEN GOTO 3660
3350 LIMIT 0,184.47,0,149.8
3360 LOCATE 7.05,119.63,L11(K),J11(K)
3370 CLIP 7.05,119.63,L11(K),U11(K)
3380 K=K+1
3390 SCALE 1,1024,0,Cmax
3400 FRAME
3410 UNCLIP
3420 LINE TYPE 1
3430 CSIZE 2.8
3440 LORG 8
3450 FOR Y=1 TO 4
3460 Ypos=Y*Cmax/4
3470 PLOT 10,Ypos,-2
3480 PLOT -9,Ypos,-1
3490 MOVE 3,Ypos
3500 LABEL PROUND(Ypos,0)
3510 NEXT Y
3520 LORG 6
3530 Ypos=Cmax/20
3540 FOR X=1 TO 8
3550 Xpos=X*128
3560 PLOT Xpos,Ypos,-2

```

```

3570 PLOT Xpos,-Ypos,-1
3580 MOVE Xpos,-Ypos+1.3
3590 LABEL Xpos
3600 NEXT X
3610 FOR J=1 TO 1023 STEP 2
3620 IF Countbin(I,J)=0 THEN GOTO 3650
3630 MOVE J,9
3640 DRAW J,Countbin(I,J)
3650 NEXT J
3660 NEXT I
3670 MOVE 512,-Ypos*5
3680 LONG 6
3690 LABEL "PLOT OF COUNT VS NUMBER OF SAMPLES PER COUNT"
3700 DUMP GRAPHICS -Ypos*10
3710 RETURN
3720 !
3730 !
3740 ISOURCE          NAM Find_vel
3750 ISOURCE !
3760 ISOURCE ! This subroutine converts raw data counts to instantaneous
3770 ISOURCE ! velocities, then sums several different products of the
3780 ISOURCE ! velocity components. All input and output data is passed
3790 ISOURCE ! through the COMMON storage area. The inputs are the raw
3800 ISOURCE ! data array (Arrayd), the Bragg shift frequency (Nub), the
3810 ISOURCE ! mixing frequencies (Numix1, Numix2, Numix3), the fringe
3820 ISOURCE ! spacings (Df1, Df2, Df3), the crossing angle of the third
3830 ISOURCE ! beam (Theta), and the number of samples in a data point (Ns).
3840 ISOURCE ! The outputs are the summations of various products of the
3850 ISOURCE ! velocity components, including U, V, W, U*U, V*V, W*W,
3860 ISOURCE ! U*V, U*W, V*W, U*U*V, U*V*V, U*U*W, U*W*W, and U*V*W.
3870 ISOURCE !
3880 ISOURCE          EXT Get_value    ! Declare subroutines stored
3890 ISOURCE          EXT Get_info     ! outside of the main program.
3900 ISOURCE          EXT Get_element
3910 ISOURCE          EXT Put_element
3920 ISOURCE          EXT Int_to_rel
3930 ISOURCE          EXT Rel_to_int
3940 ISOURCE          EXT Rel_math
3950 ISOURCE          EXT Put_value
3960 ISOURCE !
3970 ISOURCE          COM
3980 ISOURCE Data_par: INT (*)        ! Declare common variables.
3990 ISOURCE Ns_par:   INT
4000 ISOURCE Ngs_par:  INT
4010 ISOURCE Cbin_par: INT (*)
4020 ISOURCE Ran1_par: INT
4030 ISOURCE Ran2_par: INT
4040 ISOURCE Ran3_par: INT
4050 ISOURCE Sdev_par: REL
4060 ISOURCE Df1_par:  REL
4070 ISOURCE Df2_par:  REL
4080 ISOURCE Df3_par:  REL
4090 ISOURCE Theta_par: REL
4100 ISOURCE Nub_par:  REL
4110 ISOURCE Nm1x1_par: REL
4120 ISOURCE Nm1x2_par: REL
4130 ISOURCE Nm1x3_par: REL
4140 ISOURCE Su_par:    REL
4150 ISOURCE Sv_par:    REL
4160 ISOURCE Sw_par:    REL

```


4170	ISOURCE	Suu_par:	REL	
4180	ISOURCE	Suv_par:	REL	
4190	ISOURCE	Swu_par:	REL	
4200	ISOURCE	Suvu_par:	REL	
4210	ISOURCE	Suwu_par:	REL	
4220	ISOURCE	Suuu_par:	REL	
4230	ISOURCE	Suuu_par:	REL	
4240	ISOURCE	Suuu_par:	REL	
4250	ISOURCE	Suuu_par:	REL	
4260	ISOURCE	Suuu_par:	REL	
4270	ISOURCE	Suuu_par:	REL	
4280	ISOURCE	I		
4290	ISOURCE	Arrayd:	BSS 39	Reserve space for data array
4300	ISOURCE	Elementd:	EQU Arrayd+16	descriptor.
4310	ISOURCE	Array1:	BSS 4096	Reserve space for lookup tables
4320	ISOURCE	Array2:	BSS 4096	for count to velocity conversion.
4330	ISOURCE	Array3:	BSS 4096	
4340	ISOURCE	Bin_u:	BSS 1024	Reserve space for table of counts
4350	ISOURCE	Bin_v:	BSS 1024	used to generate histograms.
4360	ISOURCE	Bin_w:	BSS 1024	
4370	ISOURCE	Range1:	BSS 1	Reserve space for various input
4380	ISOURCE	Range2:	BSS 1	and output variables.
4390	ISOURCE	Range3:	BSS 1	
4400	ISOURCE	Ns:	BSS 1	
4410	ISOURCE	Df1:	BSS 4	
4420	ISOURCE	Df2:	BSS 4	
4430	ISOURCE	Df3:	BSS 4	
4440	ISOURCE	Theta:	BSS 4	
4450	ISOURCE	Nub:	BSS 4	
4460	ISOURCE	Numx1:	BSS 4	
4470	ISOURCE	Numx2:	BSS 4	
4480	ISOURCE	Numx3:	BSS 4	
4490	ISOURCE	Su:	BSS 4	
4500	ISOURCE	Sv:	BSS 4	
4510	ISOURCE	Sw:	BSS 4	
4520	ISOURCE	Suu:	BSS 4	
4530	ISOURCE	Suv:	BSS 4	
4540	ISOURCE	Swu:	BSS 4	
4550	ISOURCE	Suu:	BSS 4	
4560	ISOURCE	Suw:	BSS 4	
4570	ISOURCE	Suw:	BSS 4	
4580	ISOURCE	Suuu:	BSS 4	
4590	ISOURCE	Suuu:	BSS 4	
4600	ISOURCE	Suuu:	BSS 4	
4610	ISOURCE	Suuu:	BSS 4	
4620	ISOURCE	Suuu:	BSS 4	
4630	ISOURCE	Count:	BSS 1	Count and I are general purpose index vari-
4640	ISOURCE	I:	BSS 1	ables. Count is usually 0, 1, or 2 to denote
4650	ISOURCE	Check:	BSS 1	whether U, V, or W is being calculated.
4660	ISOURCE	Int:	BSS 1	Int, Address, and Offset are all general
4670	ISOURCE	In_2:	BSS 1	purpose storage areas.
4680	ISOURCE	Address:	BSS 1	
4690	ISOURCE	Offset:	BSS 1	
4700	ISOURCE	R1:	BSS 4	R1, R2, and R3 are the count-to-
4710	ISOURCE	R2:	BSS 4	frequency conversion factors.
4720	ISOURCE	R3:	BSS 4	
4730	ISOURCE	Xvar:	BSS 4	Xvar and Yvar are general purpose real
4740	ISOURCE	Yvar:	BSS 4	number storage areas.
4750	ISOURCE	U:	BSS 4	U, V, and W are the instantaneous velocity
4760	ISOURCE	V:	BSS 4	components.

```

4770 ISOURCE W: BSS 4
4780 ISOURCE Uu: BSS 4
4790 ISOURCE Uv: BSS 4
4800 ISOURCE Cos: BSS 4 ! Cos and Sin are the cos and sin of Theta.
4810 ISOURCE Sin: BSS 4
4820 ISOURCE Max_u: BSS 1
4830 ISOURCE Max_v: BSS 1
4840 ISOURCE Max_w: BSS 1
4850 ISOURCE Avg_u: BSS 4
4860 ISOURCE Avg_v: BSS 4
4870 ISOURCE Avg_w: BSS 4
4880 ISOURCE Dev_u: BSS 4
4890 ISOURCE Dev_v: BSS 4
4900 ISOURCE Dev_w: BSS 4
4910 ISOURCE Ngs: BSS 1
4920 ISOURCE Sdev: BSS 4
4930 ISOURCE Rad: DAT 5.729578E1
4940 ISOURCE Mill: DAT 1.E6
4950 ISOURCE One: DAT 1.
4960 ISOURCE Zero: DAT 0.
4970 ISOURCE LIT 80
4980 ISOURCE
4990 ISOURCE SUB
5000 ISOURCE Find_vel: LDA =Ns ! Get number of samples.
5010 ISOURCE LDB =Ns_par
5020 ISOURCE JSM Get_value
5030 ISOURCE LDA =Arrayd ! Get parameters of data array.
5040 ISOURCE LDB =Data_par
5050 ISOURCE JSM Get_info
5060 ISOURCE LDA =Sdev ! Get input parameters.
5070 ISOURCE LDB =Sdev_par
5080 ISOURCE JSM Get_value
5090 ISOURCE LDA =Df1
5100 ISOURCE LDB =Df1_par
5110 ISOURCE JSM Get_value
5120 ISOURCE LDA =Df2
5130 ISOURCE LDB =Df2_par
5140 ISOURCE JSM Get_value
5150 ISOURCE LDA =Df3
5160 ISOURCE LDB =Df3_par
5170 ISOURCE JSM Get_value
5180 ISOURCE LDA =Theta
5190 ISOURCE LDB =Theta_par
5200 ISOURCE JSM Get_value
5210 ISOURCE LDA =Nub
5220 ISOURCE LDB =Nub_par
5230 ISOURCE JSM Get_value
5240 ISOURCE LDA =Numix1
5250 ISOURCE LDB =Nmix1_par
5260 ISOURCE JSM Get_value
5270 ISOURCE LDA =Numix2
5280 ISOURCE LDB =Nmix2_par
5290 ISOURCE JSM Get_value
5300 ISOURCE LDA =Numix3
5310 ISOURCE LDB =Nmix3_par
5320 ISOURCE JSM Get_value
5330 ISOURCE
5340 ISOURCE ! the loop headed by Get_freq is repeated three times to get
5350 ISOURCE ! the count-to-frequency conversion factors (which depend on
5360 ISOURCE ! the range) for U,V, and W. Whenever a loop is controlled by

```

```

5370 ISOURCE | the variable "Count", the loop contains operations which are
5380 ISOURCE | the same for U,V, and W.
5390 ISOURCE |
5400 ISOURCE |
5410 ISOURCE LDA =0
5420 ISOURCE STA Count
5430 ISOURCE Get_freq: LDA Count | Get the first word of the column of
5440 ISOURCE LDB Ns | the data array which contains the
5450 ISOURCE MPY | velocity component for which we want
5460 ISOURCE STA Elementd | to get the range.
5470 ISOURCE LDA =Int
5480 ISOURCE LDB =Arrayd
5490 ISOURCE JSM Get_element
5500 ISOURCE LDA Int | Mask and rotate to get the four
5510 ISOURCE LDB =15360 | bits containing the range.
5520 ISOURCE AND B
5530 ISOURCE SAR 10
5540 ISOURCE TCA | Subtract from 15 to get the
5550 ISOURCE LDB =15 | actual range.
5560 ISOURCE ADA B
5570 ISOURCE LDB =Range1 | Store the actual range in RangeN
5580 ISOURCE ADB Count | so that we can transfer it to the
5590 ISOURCE STA B,I | main program
5600 ISOURCE LDB =1
5610 ISOURCE SZA Loopend
5620 ISOURCE Loop: SBL 1 | Use the range to find the power
5630 ISOURCE DSZ A | of two needed for the divisor.
5640 ISOURCE JMP Loop
5650 ISOURCE Loopend: STB Int
5660 ISOURCE LDA =Int | Convert the power of two into a
5670 ISOURCE STA Oper_1 | real number.
5680 ISOURCE LDA =Yvar
5690 ISOURCE STA Result
5700 ISOURCE JSM Int_to_rel
5710 ISOURCE LDA ==3.2E4 | Divide 3.2E4 by the appropriate
5720 ISOURCE LDB =Xvar | power of two, using BCD math.
5730 ISOURCE XFR 4
5740 ISOURCE STB Oper_1
5750 ISOURCE LDA =Yvar
5760 ISOURCE STA Oper_2
5770 ISOURCE LDA Count | Decide whether to put the result
5780 ISOURCE SAL 2 | in R1, R2, or R3, depending on Count
5790 ISOURCE ADA =R1
5800 ISOURCE STA Result
5810 ISOURCE LDA =2
5820 ISOURCE LDB =147155B | Now, finally, call the utility to
5830 ISOURCE JSM Rel_math | perform the division.
5840 ISOURCE ISZ Count
5850 ISOURCE LDA =3 | Increment and check Count so as
5860 ISOURCE CPA Count | to follow the loop three times.
5870 ISOURCE JMP ++2
5880 ISOURCE JMP Get_freq
5890 ISOURCE |
5900 ISOURCE LDA =Array1 | Zero out the entire count-to-
5910 ISOURCE LDB =768 | velocity conversion table so that
5920 ISOURCE Continue: CLR 16 | it must be recalculated for each
5930 ISOURCE ADA =16 | point. (This must be done if the
5940 ISOURCE DSZ B | mixing frequency or ranges are
5950 ISOURCE JMP Continue | changed between counts.)
5960 ISOURCE |

```

```

5970 ISOURCE      LDA =Su          | Set initial values of Su, Sv,
5980 ISOURCE      LDB =15          | Sv, Suu, etc. to zero.
5990 ISOURCE Clear: CLR 4
6000 ISOURCE      ADA =4
6010 ISOURCE      DSZ B
6020 ISOURCE      JMP Clear
6030 ISOURCE      |
6040 ISOURCE      LDA =Bin_u       | Clear the areas of memory which
6050 ISOURCE      LDB =192         | will be used to hold the number of
6060 ISOURCE Clear2: CLR 16        | samples per count for the histograms.
6070 ISOURCE      ADA =16
6080 ISOURCE      DSZ B
6090 ISOURCE      JMP Clear2
6100 ISOURCE      |
6110 ISOURCE      LDA =Theta       | Convert Theta from degrees to
6120 ISOURCE      STA Oper_1       | radians using the Rel_math
6130 ISOURCE      LDA =Rad         | utility.
6140 ISOURCE      STA Oper_2
6150 ISOURCE      LDA =Xvar
6160 ISOURCE      STA Result
6170 ISOURCE      LDA =2
6180 ISOURCE      LDB =147155B
6190 ISOURCE      JSM Rel_math
6200 ISOURCE      |
6210 ISOURCE      LDA =Xvar        | Find the sine and cosine of
6220 ISOURCE      STA Oper_1       | Theta, and store them in the
6230 ISOURCE      LDA =Sin         | locations Sin and Cos, respectively.
6240 ISOURCE      STA Result
6250 ISOURCE      LDA =1
6260 ISOURCE      LDB =34213B
6270 ISOURCE      JSM Rel_math
6280 ISOURCE      LDA =Cos
6290 ISOURCE      STA Result
6300 ISOURCE      LDA =1
6310 ISOURCE      LDB =34224B
6320 ISOURCE      JSM Rel_math
6330 ISOURCE      |
6340 ISOURCE      | The loop defined by Get_int calculates intermediate values
6350 ISOURCE      | used in converting counts to velocities. These values are the
6360 ISOURCE      | same for all samples in a point, so they can be calculated
6370 ISOURCE      | separately. The loop calculates (Nub-NumixN) and (Mill*DfN),
6380 ISOURCE      | where N is 1,2, and 3.
6390 ISOURCE      LDA =3
6400 ISOURCE      STA Count
6410 ISOURCE Get_int: LDA Count
6420 ISOURCE      ADA =-1
6430 ISOURCE      SAL 2
6440 ISOURCE      STA Offset
6450 ISOURCE      LDA =Nub          | Find NumixN=Nub-NumixN.
6460 ISOURCE      STA Oper_1
6470 ISOURCE      LDA =Numix1
6480 ISOURCE      ADA Offset
6490 ISOURCE      STA Oper_2
6500 ISOURCE      STA Result
6510 ISOURCE      LDA =2
6520 ISOURCE      LDB =146717B
6530 ISOURCE      JSM Rel_math
6540 ISOURCE      LDA =Df1         | Find DfN=Mill*DfN.
6550 ISOURCE      ADA Offset
6560 ISOURCE      STA Oper_1

```

```

6570 ISOURCE          STA Result
6580 ISOURCE          LDA =Mill
6590 ISOURCE          STA Oper_2
6600 ISOURCE          LDA =2
6610 ISOURCE          LDB =147037B
6620 ISOURCE          JSM Rel_math
6630 ISOURCE          DSZ Count
6640 ISOURCE          JMP Get_int
6650 ISOURCE
6660 ISOURCE | This section filters the data to remove counts which are
6670 ISOURCE | excessively far from the mean. First the section creates
6680 ISOURCE | a list of the number of times each particular count appears
6690 ISOURCE | The list consists of 3 1024 word arrays (called Bin_u, Bin_v,
6700 ISOURCE | and Bin_w) where the address of each word corresponds to the
6710 ISOURCE | count, and the value of the word indicates the number of times
6720 ISOURCE | that particular count has appeared.
6730 ISOURCE
6740 ISOURCE          LDA =0
6750 ISOURCE          STA Count
6760 ISOURCE          LDA =5
6770 ISOURCE          STA Check
6780 ISOURCE          LDA Ns
6790 ISOURCE          STA I
6800 ISOURCE
6810 ISOURCE Fill_bin: LDA Count
6820 ISOURCE          LDB Ns
6830 ISOURCE          MPY
6840 ISOURCE          ADA I
6850 ISOURCE          ADA =-1
6860 ISOURCE          STA Elementd
6870 ISOURCE          LDA =Int
6880 ISOURCE          LDB =Arrayd
6890 ISOURCE          JSM Get_element
6900 ISOURCE          LDA Int
6910 ISOURCE          LDB =1023
6920 ISOURCE          AND B
6930 ISOURCE          LDB Count
6940 ISOURCE          TCB
6950 ISOURCE          ADB Check
6960 ISOURCE          SZB Skip
6970 ISOURCE          TCA
6980 ISOURCE          ADA =1024
6990 ISOURCE Skip:   STA Int
7000 ISOURCE          LDA Count
7010 ISOURCE          LDB =1024
7020 ISOURCE          MPY
7030 ISOURCE          ADA Int
7040 ISOURCE          ADA =-1
7050 ISOURCE          ADA =Bin_u
7060 ISOURCE          ISZ A,I
7070 ISOURCE          ISZ Count
7080 ISOURCE          LDA =-3
7090 ISOURCE          ADA Count
7100 ISOURCE          SZA +=2
7110 ISOURCE          JMP Fill_bin
7120 ISOURCE          STA Count
7130 ISOURCE          DSZ I
7140 ISOURCE          JMP Fill_bin
7150 ISOURCE
7160 ISOURCE | When Sdev>0, use this section to filter the data.

```

```

7170 ISOURCE | The mean and standard deviation of the data are calculated.
7180 ISOURCE | The program then throws out all the data which are farther than
7190 ISOURCE | Sdev standard deviations from the mean.
7200 ISOURCE |
7210 ISOURCE Check_dev: LDB =Sdev
7220 ISOURCE ADB =1
7230 ISOURCE LDA B,I | If Sdev > 0, use this section to
7240 ISOURCE RZA **2 | filter the data. If not, continue
7250 ISOURCE JMP Set_begin | with the program.
7260 ISOURCE |
7270 ISOURCE LDA =0 | Get the average of each set of
7280 ISOURCE STA Count | counts using the information in
7290 ISOURCE Get_avg: LDA =Zero | the Bin_u arrays.
7300 ISOURCE LDB Count | First, set the average=0.
7310 ISOURCE SBL 2
7320 ISOURCE ADB =Avg_u
7330 ISOURCE XFR 4
7340 ISOURCE LDA Count
7350 ISOURCE ADA =1
7360 ISOURCE LDB =1024
7370 ISOURCE MPY
7380 ISOURCE ADA =Bin_u
7390 ISOURCE STA Address
7400 ISOURCE LDA =1023
7410 ISOURCE STA I
7420 ISOURCE Sum_count: LDA =I | Get a count and convert it to
7430 ISOURCE STA Oper_1 | a real number.
7440 ISOURCE LDA =Xvar
7450 ISOURCE STA Result
7460 ISOURCE JSM Int_to_real
7470 ISOURCE |
7480 ISOURCE LDA Address | Use the Bin_u arrays to find out how
7490 ISOURCE STA Oper_1 | many samples there are with that
7500 ISOURCE LDA =Yvar | count and convert that number to a
7510 ISOURCE STA Result | real number.
7520 ISOURCE JSM Int_to_real
7530 ISOURCE |
7540 ISOURCE LDA =Xvar | Multiply the count by
7550 ISOURCE STA Oper_1 | the number of times it appears in
7560 ISOURCE LDA =Yvar | the data.
7570 ISOURCE STA Oper_2
7580 ISOURCE STA Result
7590 ISOURCE LDA =2
7600 ISOURCE LDB =147037B
7610 ISOURCE JSM Rel_math
7620 ISOURCE |
7630 ISOURCE LDB Count | Keep a running sum of the product of
7640 ISOURCE SBL 2 | the count times the number of times
7650 ISOURCE ADB =Avg_u | that it appears. Store this sum in
7660 ISOURCE STB Oper_1 | the place we will eventually use to
7670 ISOURCE STB Result | store the average.
7680 ISOURCE LDA =Yvar
7690 ISOURCE STA Oper_2
7700 ISOURCE LDA =2
7710 ISOURCE LDB =146721B
7720 ISOURCE JSM Rel_math
7730 ISOURCE |
7740 ISOURCE DSZ Address
7750 ISOURCE DSZ I
7760 ISOURCE JMP Sum_count

```

'770	ISOURCE			
'780	ISOURCE	LDA =Ns		! Convert the number of samples to
'790	ISOURCE	STA Oper_1		! a real number.
'800	ISOURCE	LDA =Xvar		
'810	ISOURCE	STA Result		
'820	ISOURCE	JSM Int_to_rel		
'830	ISOURCE			
'840	ISOURCE	LDB Count		! Divide the sum stored in Avg_u by
'850	ISOURCE	SBL 2		! the number of samples to get the
'860	ISOURCE	ADB =Avg_u		! average value of the counts.
'870	ISOURCE	STB Oper_1		
'880	ISOURCE	STB Result		
'890	ISOURCE	LDB =Xvar		
'900	ISOURCE	STB Oper_2		
'910	ISOURCE	LDA =2		
'920	ISOURCE	LDB =147155B		
'930	ISOURCE	JSM Rel_math		
'940	ISOURCE			
'950	ISOURCE	LDA Count		
'960	ISOURCE	ADA =1		
'970	ISOURCE	STA Count		
'980	ISOURCE	ADA =-3		
'990	ISOURCE	SZA ++2		
8000	ISOURCE	JMP Get_avg		
8010	ISOURCE			
8020	ISOURCE	LDA =0		! Get the standard deviation of each
8030	ISOURCE	STA Count		! set of counts using the average and
8040	ISOURCE	Get_sdev: LDA =Zero		! the information in the Bin_u arrays.
8050	ISOURCE	LDB Count		! First, set the standard deviation =0.
8060	ISOURCE	SBL 2		
8070	ISOURCE	STB Offset		
8080	ISOURCE	ADB =Dev_u		
8090	ISOURCE	XFR 4		
8100	ISOURCE	LDA Offset		
8110	ISOURCE	ADA =Avg_u		
8120	ISOURCE	STA Int		
8130	ISOURCE	LDA Count		
8140	ISOURCE	ADA =1		
8150	ISOURCE	LDB =1024		
8160	ISOURCE	MPY		
8170	ISOURCE	ADA =Bin_u		
8180	ISOURCE	STA Address		
8190	ISOURCE	LDA =1023		
8200	ISOURCE	STA I		
8210	ISOURCE			
8220	ISOURCE	Dev_count: LDA =I		! Get a count and convert it to a
8230	ISOURCE	STA Oper_1		! real number.
8240	ISOURCE	LDA =Xvar		
8250	ISOURCE	STA Result		
8260	ISOURCE	JSM Int_to_rel		
8270	ISOURCE			
8280	ISOURCE	LDA =Xvar		! Subtract the average count from the
8290	ISOURCE	STA Oper_1		! count we just got.
8300	ISOURCE	LDA Int		
8310	ISOURCE	STA Oper_2		
8320	ISOURCE	LDA =Yvar		
8330	ISOURCE	STA Result		
8340	ISOURCE	LDA =2		
8350	ISOURCE	LDB =146717B		
8360	ISOURCE	JSM Rel_math		

8370	ISOURCE		
8380	ISOURCE	LDA =Yvar	Square the difference between the
8390	ISOURCE	STA Oper_1	average and the count we got to
8400	ISOURCE	STA Oper_2	get the deviation from the mean.
8410	ISOURCE	LDA =Xvar	
8420	ISOURCE	STA Result	
8430	ISOURCE	LDA =2	
8440	ISOURCE	LDB =147037B	
8450	ISOURCE	JSM Rel_math	
8460	ISOURCE		
8470	ISOURCE	LDA Address	Use the Bin_u arrays to find out
8480	ISOURCE	STA Oper_1	how many counts there are with the
8490	ISOURCE	LDA =Yvar	value we have chosen, and convert
8500	ISOURCE	STA Result	that number to a real number.
8510	ISOURCE	JSM Int_to_real	
8520	ISOURCE		
8530	ISOURCE	LDA =Xvar	Multiply the number of counts by
8540	ISOURCE	STA Oper_1	the deviation from the mean.
8550	ISOURCE	LDA =Yvar	
8560	ISOURCE	STA Oper_2	
8570	ISOURCE	STA Result	
8580	ISOURCE	LDA =2	
8590	ISOURCE	LDB =147037B	
8600	ISOURCE	JSM Rel_math	
8610	ISOURCE		
8620	ISOURCE	LDA =Yvar	Add the product to Dev_u, which
8630	ISOURCE	STA Oper_1	currently contains a running sum of
8640	ISOURCE	LDA =Dev_u	the number of counts times the
8650	ISOURCE	ADA Offset	deviation from the mean of each
8660	ISOURCE	STA Oper_2	count.
8670	ISOURCE	STA Result	
8680	ISOURCE	LDA =2	
8690	ISOURCE	LDB =146721B	
8700	ISOURCE	JSM Rel_math	
8710	ISOURCE		
8720	ISOURCE	DSZ Address	
8730	ISOURCE	DSZ I	
8740	ISOURCE	JMP Dev_count	
8750	ISOURCE		
8760	ISOURCE	LDA =Ns	Convert the number of samples to
8770	ISOURCE	STA Oper_1	a real number.
8780	ISOURCE	LDA =Xvar	
8790	ISOURCE	STA Result	
8800	ISOURCE	JSM Int_to_real	
8810	ISOURCE		
8820	ISOURCE	LDB Offset	Divide the sum stored in Dev_u by
8830	ISOURCE	ADB =Dev_u	the number of samples to get the
8840	ISOURCE	STB Oper_1	variance of the counts.
8850	ISOURCE	STB Result	
8860	ISOURCE	LDB =Xvar	
8870	ISOURCE	STB Oper_2	
8880	ISOURCE	LDA =2	
8890	ISOURCE	LDB =147155B	
8900	ISOURCE	JSM Rel_math	
8910	ISOURCE		
8920	ISOURCE	LDA =Dev_u	Take the square root of the
8930	ISOURCE	ADA Offset	variance to get the standard
8940	ISOURCE	STA Oper_1	deviation.
8950	ISOURCE	STA Result	
8960	ISOURCE	LDA =1	

8970	ISOURCE	LDB =31450B	
8980	ISOURCE	JSM Rel_math	
8990	ISOURCE		
9000	ISOURCE	LDA Count	
9010	ISOURCE	ADA =1	
9020	ISOURCE	STA Count	
9030	ISOURCE	ADA =-3	
9040	ISOURCE	SZA **2	
9050	ISOURCE	JMP Get_sdev	
9060	ISOURCE		
9070	ISOURCE	LDA =0	! Now use the standard deviation to
9080	ISOURCE	STA Count	! filter out all the counts whose
9090	ISOURCE	Dev_filt:n: LDA Count	! value is more than Sdev standard
9100	ISOURCE	ADA =1	! deviations away from the mean.
9110	ISOURCE	LDB =1024	
9120	ISOURCE	MPY	
9130	ISOURCE	ADA =Bin_u	
9140	ISOURCE	STA Address	
9150	ISOURCE	LDA =1023	
9160	ISOURCE	STA I	
9170	ISOURCE		
9180	ISOURCE	LDA Count	
9190	ISOURCE	SAL 2	
9200	ISOURCE	ADA =Dev_u	
9210	ISOURCE	STA Oper_1	
9220	ISOURCE	LDA =Sdev	
9230	ISOURCE	STA Oper_2	
9240	ISOURCE	LDA =Xvar	
9250	ISOURCE	STA Result	
9260	ISOURCE	LDA =2	
9270	ISOURCE	LDB =147037B	
9280	ISOURCE	JSM Rel_math	
9290	ISOURCE		
9300	ISOURCE	LDA =Xvar	
9310	ISOURCE	STA Oper_1	
9320	ISOURCE	LDA =Int	
9330	ISOURCE	STA Result	
9340	ISOURCE	JSM Rel_to_int	
9350	ISOURCE		
9360	ISOURCE	LDA Count	
9370	ISOURCE	SAL 2	
9380	ISOURCE	ADA =Avg_u	
9390	ISOURCE	STA Oper_1	
9400	ISOURCE	LDA =Offset	
9410	ISOURCE	STA Result	
9420	ISOURCE	JSM Rel_to_int	
9430	ISOURCE		
9440	ISOURCE	LDB Offset	
9450	ISOURCE	TCB	
9460	ISOURCE	Filtr_dev: LDA I	
9470	ISOURCE	ADA B	
9480	ISOURCE	SAM **2	
9490	ISOURCE	TCA	
9500	ISOURCE	ADA Int	
9510	ISOURCE	SAP **3	
9520	ISOURCE	LDA Address	
9530	ISOURCE	CLR I	
9540	ISOURCE	DSZ Address	
9550	ISOURCE	DSZ I	
9560	ISOURCE	JMP Filtr_dev	

```

9570 ISOURCE !
9580 ISOURCE      ISZ Count      ! Go through this whole process
9590 ISOURCE      LDA #-3        ! of checking the deviation and
9600 ISOURCE      ADA Count      ! filtering the data three times,
9610 ISOURCE      SZA **2        ! once for each channel.
9620 ISOURCE      JMP Dev_filtn
9630 ISOURCE !
9640 ISOURCE ! The loop Begin is performed three times, once for U, V,
9650 ISOURCE ! and W. Each time through the data array is read, a count (a raw
9660 ISOURCE ! datum) is taken from it and converted into a velocity. The vel-
9670 ISOURCE ! ocity is stored in U, V, or W depending on whether this is the
9680 ISOURCE ! first, second, or third iteration of the loop. The first time a
9690 ISOURCE ! particular count is encountered, the velocity corresponding to
9700 ISOURCE ! it is calculated using the intermediate values found in Get_int,
9710 ISOURCE ! and the velocity is stored in a table. If that count is found
9720 ISOURCE ! again in the data array, the corresponding velocity is looked
9730 ISOURCE ! up rather than being calculated again.
9740 ISOURCE !
9750 ISOURCE Set_begin: LDA #0
9760 ISOURCE      STA Count
9770 ISOURCE      LDA Ns
9780 ISOURCE      STA I
9790 ISOURCE      STA Ngs
9800 ISOURCE !
9810 ISOURCE Begin:   LDA Count
9820 ISOURCE      STA B
9830 ISOURCE      SBL 2
9840 ISOURCE      STB Offset
9850 ISOURCE      LD, Ns      ! Figure out which element of the
9860 ISOURCE      MPY          ! data array we want to pick up.
9870 ISOURCE      ADA I
9880 ISOURCE      ADA #-1
9890 ISOURCE      STA Elementd
9900 ISOURCE      LDA =Int     ! Get a raw datum from the data array.
9910 ISOURCE      LDB =Arrayd
9920 ISOURCE      JSM Get_element
9930 ISOURCE      LDA Int
9940 ISOURCE      LDB =1023    ! Strip off the first six
9950 ISOURCE      AND B        ! bits of the raw data word.
9960 ISOURCE      LDB Count    ! See if Count = Check.
9970 ISOURCE      TCB
9980 ISOURCE      ADB Check
9990 ISOURCE      SZB Straight ! If true, use the modified data
10000 ISOURCE      TCA        ! word as an index. If not, use
10010 ISOURCE      ADA =1024   ! 1024 minus the data word.
10020 ISOURCE Straight: STA Int ! Store the count we have gotten.
10030 ISOURCE !
10040 ISOURCE      LDA Count    ! Now look in the appropriate part
10050 ISOURCE      LDB =1024    ! of the arrays created by the filter
10060 ISOURCE      MPY          ! section.
10070 ISOURCE      ADA Int
10080 ISOURCE      ADA #-1
10090 ISOURCE      ADA =Bin_u
10100 ISOURCE      LDB A,I
10110 ISOURCE      RZB Goodcount ! If the word corresponding to the
10120 ISOURCE      STB Count     ! count is zero, the count was
10130 ISOURCE      DSZ Ngs       ! filtered out and should be ignored.
10140 ISOURCE      DSZ I        ! Ignore these three counts, go back
10150 ISOURCE      JMP Begin    ! and do the next three.
10160 ISOURCE !

```

10170	ISOURCE	Goodcount:	LDA Count	Now, use Count to find out
10180	ISOURCE		LDB =4096	which lookup table array
10190	ISOURCE		MPY	we want to use, and use the
10200	ISOURCE		LDB Int	count we got from the data array
10210	ISOURCE		ADB =-1	to find exactly where in the
10220	ISOURCE		SBL 2	table we want to go.
10230	ISOURCE		ADA B	
10240	ISOURCE		ADA =1	
10250	ISOURCE		ADA =Array1	
10260	ISOURCE		STA Address	
10270	ISOURCE		LDB =Int2	
10280	ISOURCE		XFR 1	
10290	ISOURCE		LDA Int2	If that table entry is zero,
10300	ISOURCE		SZA Calculate	calculate a velocity for it.
10310	ISOURCE		JMP Over	
10320	ISOURCE	Calculate:	LDA =Int	
10330	ISOURCE		STA Oper_1	Convert the count into
10340	ISOURCE		LDA =Yvar	a real number.
10350	ISOURCE		STA Result	
10360	ISOURCE		JSM Int_to_rel	
10370	ISOURCE		STB Oper_2	
10380	ISOURCE		LDA =R1	Divide the range we found
10390	ISOURCE		ADA Offset	earlier by the count to get a
10400	ISOURCE		STA Oper_1	frequency.
10410	ISOURCE		LDA =Xvar	
10420	ISOURCE		STA Result	
10430	ISOURCE		LDA =2	
10440	ISOURCE		LDB =147155B	
10450	ISOURCE		JSM Rel_math	
10460	ISOURCE			
10470	ISOURCE		LDA =Numx1	Find (Nub-NumxN)-FrequencyN.
10480	ISOURCE		ADA Offset	
10490	ISOURCE		STA Oper_1	
10500	ISOURCE		LDA =Xvar	
10510	ISOURCE		STA Oper_2	
10520	ISOURCE		LDA =Yvar	
10530	ISOURCE		STA Result	
10540	ISOURCE		LDA =2	
10550	ISOURCE		LDB =146717B	
10560	ISOURCE		JSM Rel_math	
10570	ISOURCE			
10580	ISOURCE		LDB Count	If we are calculating U, reverse
10590	ISOURCE		ADB =-1	the sign of ((Nub-NumxN)-FrequencyN)
10600	ISOURCE		SBM **2	so as to reverse the sign of U.
10610	ISOURCE		JMP Samesign	Leave V and Wv alone.
10620	ISOURCE		LDA =Zero	
10630	ISOURCE		STA Oper_1	
10640	ISOURCE		LDA Result	
10650	ISOURCE		STA Oper_2	
10660	ISOURCE		LDA =Xvar	
10670	ISOURCE		STA Result	
10680	ISOURCE		LDA =2	
10690	ISOURCE		LDB =146717B	
10700	ISOURCE		JSM Rel_math	
10710	ISOURCE	Samesign:	NOP	
10720	ISOURCE			
10730	ISOURCE		LDA Result	Find Velocity=((Nub-NumxN)
10740	ISOURCE		STA Oper_1	-FrequencyN)*(Mill*DfN)
10750	ISOURCE		LDA =Df1	and store in a place in the
10760	ISOURCE		ADA Offs=+	lookup table corresponding to

```

10770 ISOURCE      STA Oper_2      | the data count.
10780 ISOURCE      LDA Address
10790 ISOURCE      ADA =-1
10800 ISOURCE      STA Result
10810 ISOURCE      LDA =2
10820 ISOURCE      LDB =147037B
10830 ISOURCE      JSM Rel_math
10840 ISOURCE      |
10850 ISOURCE      Over:      LDA Address      | Transfer the velocity from
10860 ISOURCE      ADA =-1      | the lookup table to U, V, or W,
10870 ISOURCE      LDB =U      | as appropriate.
10880 ISOURCE      ADB Offset
10890 ISOURCE      XFR 4
10900 ISOURCE      |
10910 ISOURCE      ISZ Count
10920 ISOURCE      LDA =-3      | Have U, V, and W all
10930 ISOURCE      ADA Count      | been calculated?
10940 ISOURCE      SZA +=2      | If not, go back again.
10950 ISOURCE      JMP Begin
10960 ISOURCE      STA Count      | If so, set Count = 0.
10970 ISOURCE      |
10980 ISOURCE      | Now we convert the W we have obtained (which is measured at an
10990 ISOURCE      | angle Theta to the V-axis) to the W we want (which should
11000 ISOURCE      | be measured at an angle of 90 degrees to the V-axis).
11010 ISOURCE      | Thus find  $W = (WU - V * \cos(\theta)) / \sin(\theta)$ .
11020 ISOURCE      |
11030 ISOURCE      LDA =V
11040 ISOURCE      STA Oper_1
11050 ISOURCE      LDA =Cos
11060 ISOURCE      STA Oper_2
11070 ISOURCE      LDA =Xvar
11080 ISOURCE      STA Result
11090 ISOURCE      LDA =2
11100 ISOURCE      LDB =147037B
11110 ISOURCE      JSM Rel_math
11120 ISOURCE      LDA Result
11130 ISOURCE      STA Oper_2
11140 ISOURCE      LDA =W
11150 ISOURCE      STA Oper_1
11160 ISOURCE      LDA =Yvar
11170 ISOURCE      STA Result
11180 ISOURCE      LDA =2
11190 ISOURCE      LDB =146717B
11200 ISOURCE      JSM Rel_math
11210 ISOURCE      LDA Result
11220 ISOURCE      STA Oper_1
11230 ISOURCE      LDA =Sin
11240 ISOURCE      STA Oper_2
11250 ISOURCE      LDA =W
11260 ISOURCE      STA Result
11270 ISOURCE      LDA =2
11280 ISOURCE      LDB =147155B
11290 ISOURCE      JSM Rel_math
11300 ISOURCE      |
11310 ISOURCE      | Now take running sums of U, V, W, and several products
11320 ISOURCE      | of these velocities. The sums are taken using the utility
11330 ISOURCE      | "Add". The sums are calculated in an unusual sequence in
11340 ISOURCE      | order to reduce the number of program steps needed to cal-
11350 ISOURCE      | culate them.
11360 ISOURCE      |

```

11370	ISOURCE	LDA =Su	Find Su=Su+U
11380	ISOURCE	LDB =U	
11390	ISOURCE	JSM Add	
11400	ISOURCE		
11410	ISOURCE	LDA Oper_2	Find U*U
11420	ISOURCE	STA Oper_1	
11430	ISOURCE	LDA =Uu	
11440	ISOURCE	STA Result	
11450	ISOURCE	LDA =2	
11460	ISOURCE	LDB =147037B	
11470	ISOURCE	JSM Rel_math	
11480	ISOURCE		
11490	ISOURCE	LDA =Suu	Find Suu=Suu+(U*U)
11500	ISOURCE	LDB Result	
11510	ISOURCE	JSM Add	
11520	ISOURCE		
11530	ISOURCE	LDA =V	Find U*U*V
11540	ISOURCE	STA Oper_1	
11550	ISOURCE	LDA =Xvar	
11560	ISOURCE	STA Result	
11570	ISOURCE	LDA =2	
11580	ISOURCE	LDB =147037B	
11590	ISOURCE	JSM Rel_math	
11600	ISOURCE		
11610	ISOURCE	LDA =Suuv	Find Suuv=Suuv+(U*U*V)
11620	ISOURCE	LDB Result	
11630	ISOURCE	JSM Add	
11640	ISOURCE		
11650	ISOURCE	LDA =Uu	Find U*U*W
11660	ISOURCE	STA Oper_1	
11670	ISOURCE	LDA =W	
11680	ISOURCE	STA Oper_2	
11690	ISOURCE	LDA =Xvar	
11700	ISOURCE	STA Result	
11710	ISOURCE	LDA =2	
11720	ISOURCE	LDB =147037B	
11730	ISOURCE	JSM Rel_math	
11740	ISOURCE		
11750	ISOURCE	LDA =Suuw	Find Suuw=Suuw+(U*U*W)
11760	ISOURCE	LDB Result	
11770	ISOURCE	JSM Add	
11780	ISOURCE		
11790	ISOURCE	LDA =Su	Find Sv=Su+V
11800	ISOURCE	LDB =V	
11810	ISOURCE	JSM Add	
11820	ISOURCE		
11830	ISOURCE	LDA =U	Find U*V
11840	ISOURCE	STA Oper_1	
11850	ISOURCE	LDA =Uv	
11860	ISOURCE	STA Result	
11870	ISOURCE	LDA =2	
11880	ISOURCE	LDB =147037B	
11890	ISOURCE	JSM Rel_math	
11900	ISOURCE		
11910	ISOURCE	LDA =Suv	Find Suv=Suv+(U*V)
11920	ISOURCE	LDB Result	
11930	ISOURCE	JSM Add	
11940	ISOURCE		
11950	ISOURCE	LDA =V	Find U*V*V
11960	ISOURCE	STA Oper_1	

11970	ISOURCE	LDA =Xvar	
11980	ISOURCE	STA Result	
11990	ISOURCE	LDA =2	
12000	ISOURCE	LDB =147037B	
12010	ISOURCE	JSM Rel_math	
12020	ISOURCE		
12030	ISOURCE	LDA =Suuv	! Find Suuv=Suuv+(U*V*V)
12040	ISOURCE	LDB Result	
12050	ISOURCE	JSM Add	
12060	ISOURCE		
12070	ISOURCE	LDA =Uv	! Find U*V*W
12080	ISOURCE	STA Oper_1	
12090	ISOURCE	LDA =W	
12100	ISOURCE	STA Oper_2	
12110	ISOURCE	LDA =Xvar	
12120	ISOURCE	STA Result	
12130	ISOURCE	LDA =2	
12140	ISOURCE	LDB =147037B	
12150	ISOURCE	JSM Rel_math	
12160	ISOURCE		
12170	ISOURCE	LDA =Suuv	! Find Suuv=Suuv+(U*V*W)
12180	ISOURCE	LDB Result	
12190	ISOURCE	JSM Add	
12200	ISOURCE		
12210	ISOURCE	LDA =V	! Find V*V
12220	ISOURCE	STA Oper_1	
12230	ISOURCE	STA Oper_2	
12240	ISOURCE	LDA =Xvar	
12250	ISOURCE	STA Result	
12260	ISOURCE	LDA =2	
12270	ISOURCE	LDB =147037B	
12280	ISOURCE	JSM Rel_math	
12290	ISOURCE		
12300	ISOURCE	LDA =Suuv	! Find Suuv=Suuv+(V*V)
12310	ISOURCE	LDB Result	
12320	ISOURCE	JSM Add	
12330	ISOURCE		
12340	ISOURCE	LDA =Sw	! Find Sw=Sw+W
12350	ISOURCE	LDB =W	
12360	ISOURCE	JSM Add	
12370	ISOURCE		
12380	ISOURCE	LDA =U	! Find U*W
12390	ISOURCE	STA Oper_1	
12400	ISOURCE	LDA =Xvar	
12410	ISOURCE	STA Result	
12420	ISOURCE	LDA =2	
12430	ISOURCE	LDB =147037B	
12440	ISOURCE	JSM Rel_math	
12450	ISOURCE		
12460	ISOURCE	LDA =Suw	! Find Suw=Suw+(U*W)
12470	ISOURCE	LDB Result	
12480	ISOURCE	JSM Add	
12490	ISOURCE		
12500	ISOURCE	LDA =W	! Find V*W
12510	ISOURCE	STA Oper_1	
12520	ISOURCE	LDA =V	
12530	ISOURCE	STA Oper_2	
12540	ISOURCE	LDA =Xvar	
12550	ISOURCE	STA Result	
12560	ISOURCE	LDA =2	

```

12570 ISOURCE      LDB =147037B
12580 ISOURCE      JSM Rel_math
12590 ISOURCE      '
12600 ISOURCE      LDA =Suv      ' Find Suu=Suv+(V*W)
12610 ISOURCE      LDB Result
12620 ISOURCE      JSM Add
12630 ISOURCE      '
12640 ISOURCE      LDA =W      ' Find W*W
12650 ISOURCE      STA Oper_1
12660 ISOURCE      STA Oper_2
12670 ISOURCE      LDA =Xvar
12680 ISOURCE      STA Result
12690 ISOURCE      LDA =2
12700 ISOURCE      LDB =147037B
12710 ISOURCE      JSM Rel_math
12720 ISOURCE      '
12730 ISOURCE      LDA =Suw      ' Find Suw=Suw+(W*W)
12740 ISOURCE      LDB Result
12750 ISOURCE      JSM Add
12760 ISOURCE      '
12770 ISOURCE      LDA =U      ' Find U*W*W
12780 ISOURCE      STA Oper_1
12790 ISOURCE      LDA =Uu
12800 ISOURCE      STA Result
12810 ISOURCE      LDA =2
12820 ISOURCE      LDB =147037B
12830 ISOURCE      JSM Rel_math
12840 ISOURCE      '
12850 ISOURCE      LDA =Suww      ' Find Suww=Suww+(U*W*W)
12860 ISOURCE      LDB Result
12870 ISOURCE      JSM Add
12880 ISOURCE      '
12890 ISOURCE      DSZ I      ' Continue to calculate running
12900 ISOURCE      JMP Begin      ' sums until out of samples.
12910 ISOURCE      '
12920 ISOURCE      ' Now place the finished sums in the COMMON region so that
12930 ISOURCE      ' the BASIC program has access to them, and then return to
12940 ISOURCE      ' the BASIC program.
12950 ISOURCE      Replace: LDA =Range1
12960 ISOURCE      LDB =Ran1_par
12970 ISOURCE      JSM Put_value
12980 ISOURCE      LDA =Range2
12990 ISOURCE      LDB =Ran2_par
13000 ISOURCE      JSM Put_value
13010 ISOURCE      LDA =Range3
13020 ISOURCE      LDB =Ran3_par
13030 ISOURCE      JSM Put_value
13040 ISOURCE      LDA =Su
13050 ISOURCE      LDB =Su_par
13060 ISOURCE      JSM Put_value
13070 ISOURCE      LDA =Su
13080 ISOURCE      LDB =Su_par
13090 ISOURCE      JSM Put_value
13100 ISOURCE      LDA =Su
13110 ISOURCE      LDB =Su_par
13120 ISOURCE      JSM Put_value
13130 ISOURCE      LDA =Suu
13140 ISOURCE      LDB =Suu_par
13150 ISOURCE      JSM Put_value
13160 ISOURCE      LDA =Suv

```

13170	ISOURCE	LDB =Suv_par	
13180	ISOURCE	JSM Put_value	
13190	ISOURCE	LDA =Suw	
13200	ISOURCE	LDB =Suw_par	
13210	ISOURCE	JSM Put_value	
13220	ISOURCE	LDA =Suw	
13230	ISOURCE	LDB =Suv_par	
13240	ISOURCE	JSM Put_value	
13250	ISOURCE	LDA =Suw	
13260	ISOURCE	LDB =Suw_par	
13270	ISOURCE	JSM Put_value	
13280	ISOURCE	LDA =Suw	
13290	ISOURCE	LDB =Suw_par	
13300	ISOURCE	JSM Put_value	
13310	ISOURCE	LDA =Suuv	
13320	ISOURCE	LDB =Suuv_par	
13330	ISOURCE	JSM Put_value	
13340	ISOURCE	LDA =Suuv	
13350	ISOURCE	LDB =Suuv_par	
13360	ISOURCE	JSM Put_value	
13370	ISOURCE	LDA =Suuw	
13380	ISOURCE	LDB =Suuw_par	
13390	ISOURCE	JSM Put_value	
13400	ISOURCE	LDA =Suuw	
13410	ISOURCE	LDB =Suuw_par	
13420	ISOURCE	JSM Put_value	
13430	ISOURCE	LDA =Suuv	
13440	ISOURCE	LDB =Suuv_par	
13450	ISOURCE	JSM Put_value	
13460	ISOURCE	LDA =Ngs	
13470	ISOURCE	LDB =Ngs_par	
13480	ISOURCE	JSM Put_value	
13490	ISOURCE		
13500	ISOURCE	LDA =Bin_u	! Transfer the contents of Bin_n
13510	ISOURCE	LDB =192	! to the common area without using
13520	ISOURCE	STB I	! the slow HP-supplied external
13530	ISOURCE	LDB =16	! subroutines.
13540	ISOURCE	STB Count	
13550	ISOURCE	LDB Cbin_par	
13560	ISOURCE	Transfer: XFR 16	
13570	ISOURCE	ADA Count	
13580	ISOURCE	ADB Count	
13590	ISOURCE	DSZ I	
13600	ISOURCE	JMP Transfer	
13610	ISOURCE	RET 1	
13620	ISOURCE		
13630	ISOURCE	LIT 200	
13640	ISOURCE		
13650	ISOURCE		! The utility "Add" is used to add up the running sums.
13660	ISOURCE		
13670	ISOURCE	Add: IS7 Utilcount	
13680	ISOURCE	STA Oper_1	
13690	ISOURCE	STB Oper_2	
13700	ISOURCE	STA Result	
13710	ISOURCE	LDA =2	
13720	ISOURCE	LDB =146721B	
13730	ISOURCE	JSM Rel_math	
13740	ISOURCE	DSZ Utilcount	
13750	ISOURCE	RET 1	
13760	ISOURCE	JSM Utilend	

13770 ISOURCE I
13780 ISOURCE

END Find_vel

ACKNOWLEDGEMENTS

We are grateful to Dr. Oktay Ozcan for help in the initial design of the system and to Dr. Dennis Johnson for many helpful comments. This work was supported by the Fluid Dynamics Research Branch, NASA Ames Research Center under Grant NCC-2-294.

REFERENCES

1. Bell, J. H., Rodman, L. C., and Mehta, R. D., "Aspects of Design and Performance of a 3-Component LDV System", Paper presented at the 11th ICIASF Meeting on Instrumentation, Stanford University, Aug. 26-28, 1985.
2. Westphal, R. V., and Mehta, R. D., "Crossed Hot-Wire Data Acquisition and Reduction System", NASA TM-85871, 1984.
3. LASER DOPPLER VELOCIMETER INSTRUCTION MANUAL, System 9100-7, TSI Incorp., St. Paul, MN, August 1980.
4. FREQUENCY SHIFT SYSTEM INSTRUCTION MANUAL, Model 9180, Revision B, TSI Incorp., St. Paul, MN, August 1979.
5. OPERATOR'S MANUAL FOR NASA 3D-LDV COMPUTER INTERFACE, NASA Ames Research Center, Writing Assoc., Sunnyvale, CA, July 1, 1982.
6. Mehta, R. D., "An Experimental Study of a Vortex/Mixing Layer Interaction", AIAA Paper 84-1543, 1984.

TABLE 1
MACRODYNE RANGE SETTINGS

MANTISA	1	500	1024
RANGE			
0	32 GHz	64 MHz	31.28 MHz
1	16 "	32 "	15.64 "
2	8 "	16 "	7.20 "
3	4 "	8 "	3.91 "
4	2 "	4 "	1.95 "
5	1 "	2 "	977 kHz
6	500 MHz	1 "	488 "
7	250 "	500 kHz	244 "
8	125 "	250 "	122 "
9	62.5 "	125 "	61 "
10	31.25 "	62.5 "	30 "
11	15.62 "	31.25 "	15 "
12	7.81 "	15.62 "	7.6 "
13	3.90 "	7.81 "	3.8 "
14	1.95 "	3.90 "	1.9 "

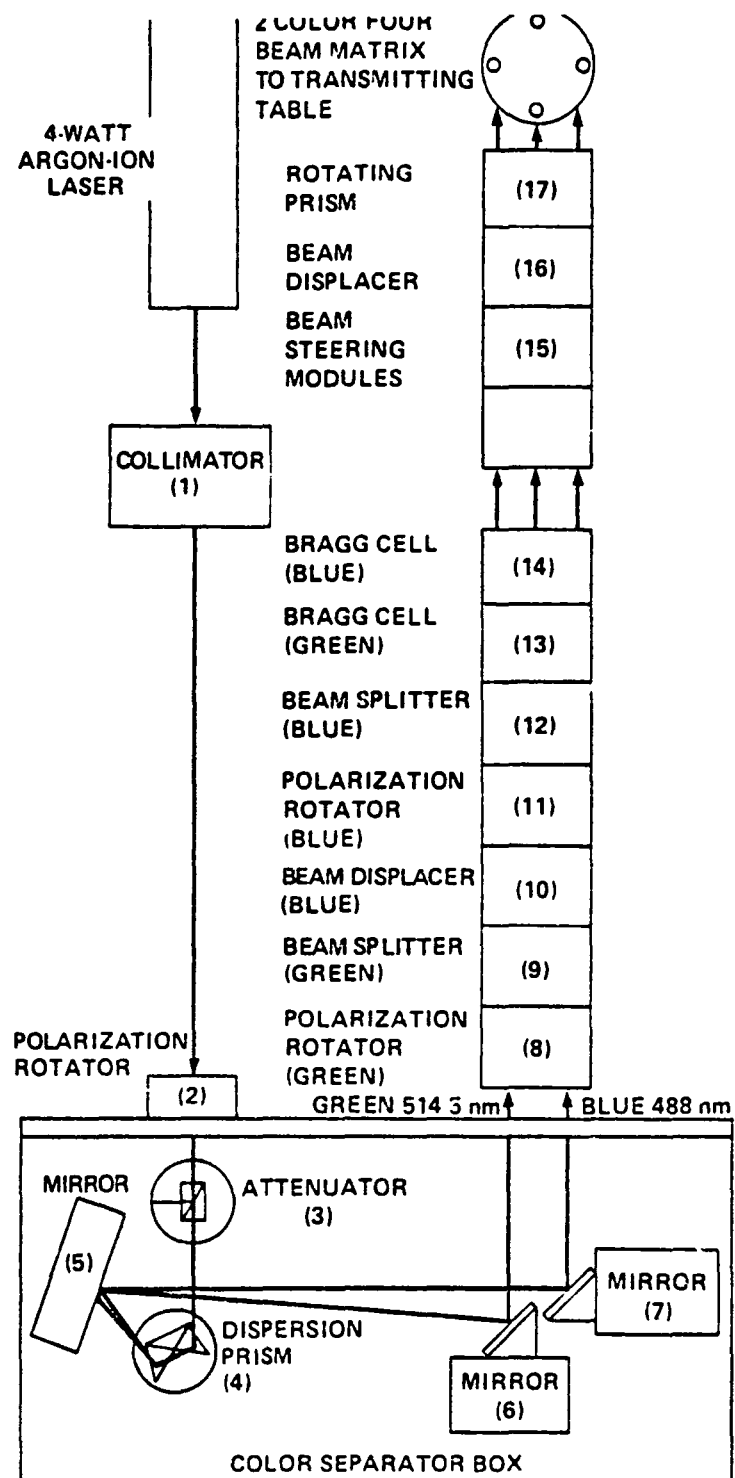
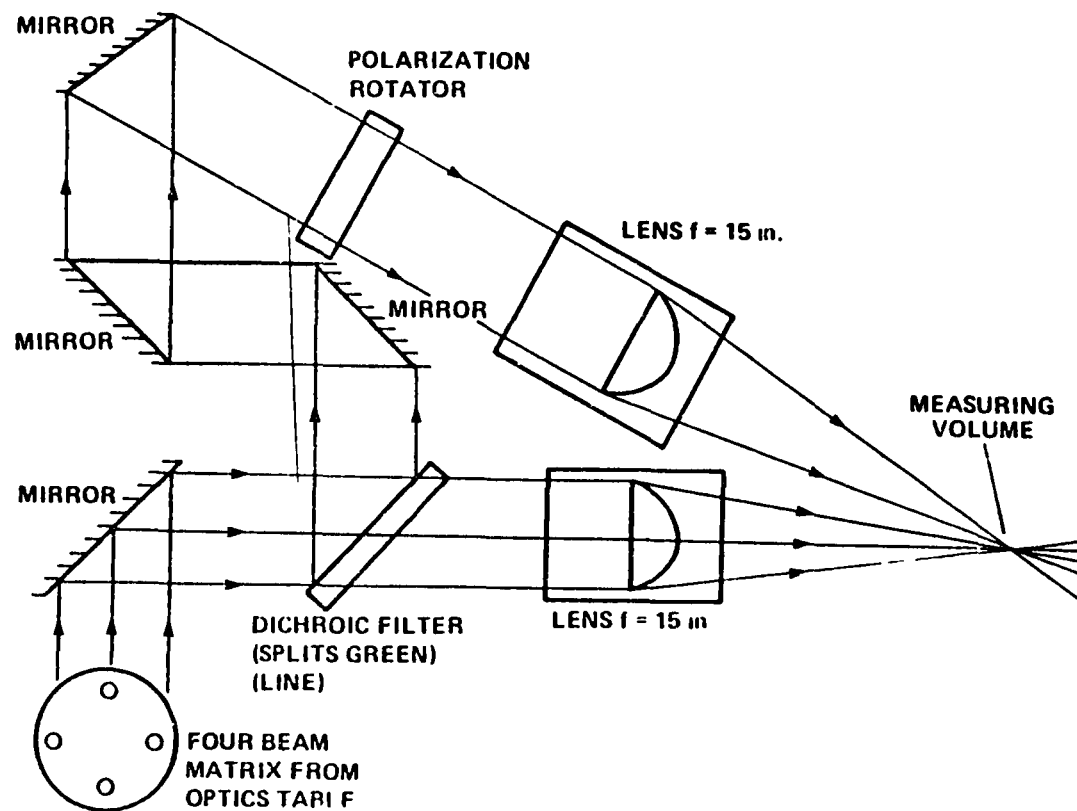


Fig. 1 Schematic of the optics table layout.

Fig. 2 Schematic of the transmitting optics.



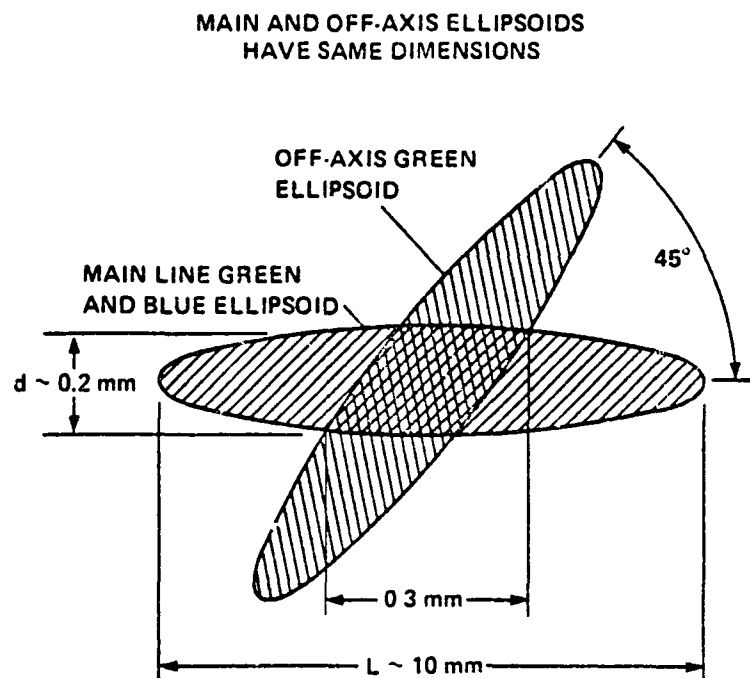


Fig. 3 Details of the probe volumes in the 3-D system.

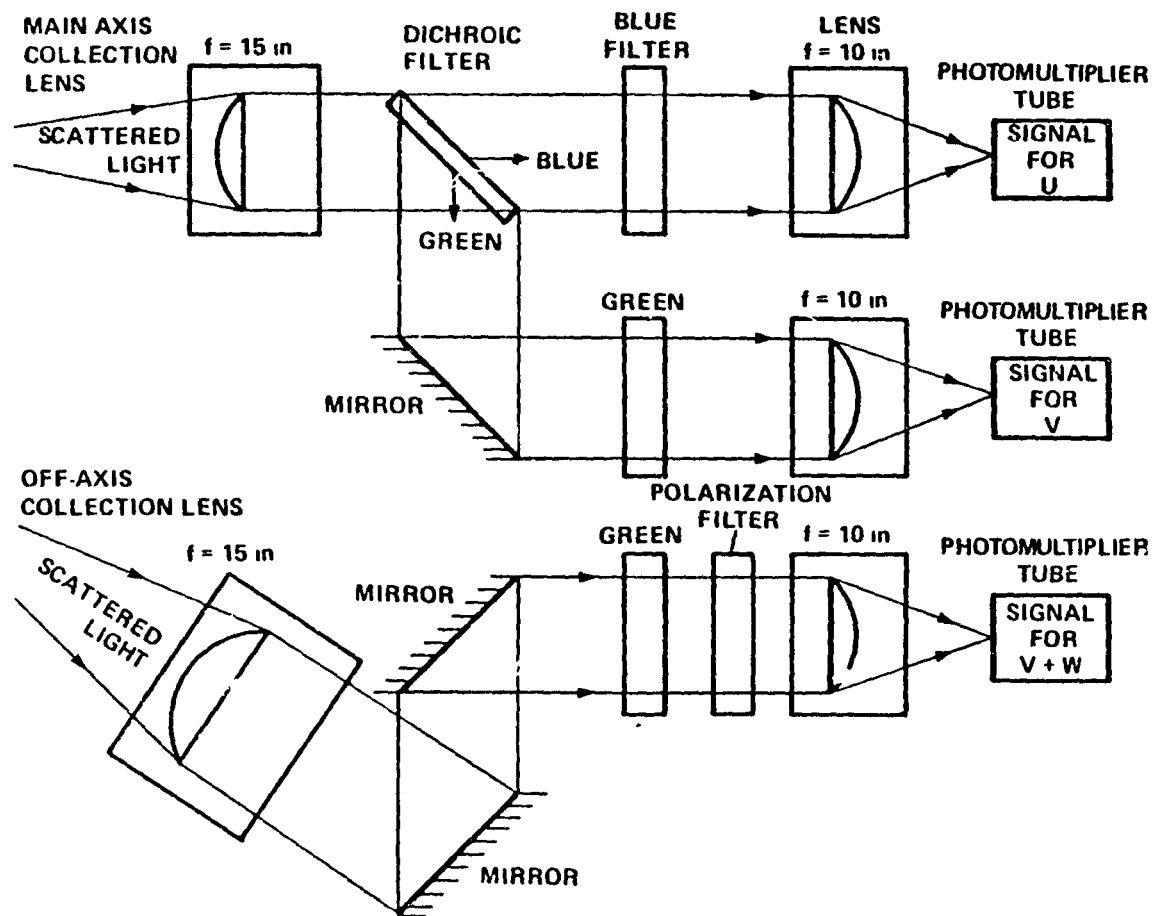


Fig. 4 Schematic of the receiving optics.

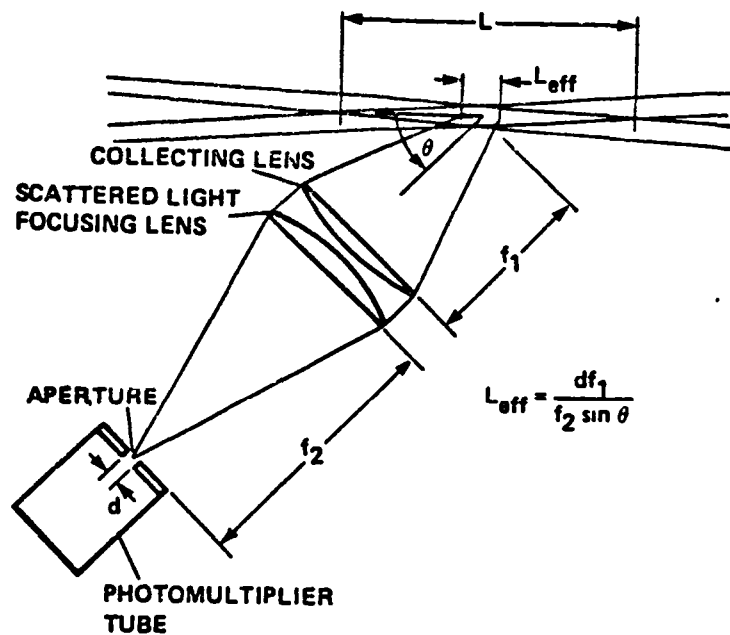


Fig. 5 Evaluation of the effective probe length.

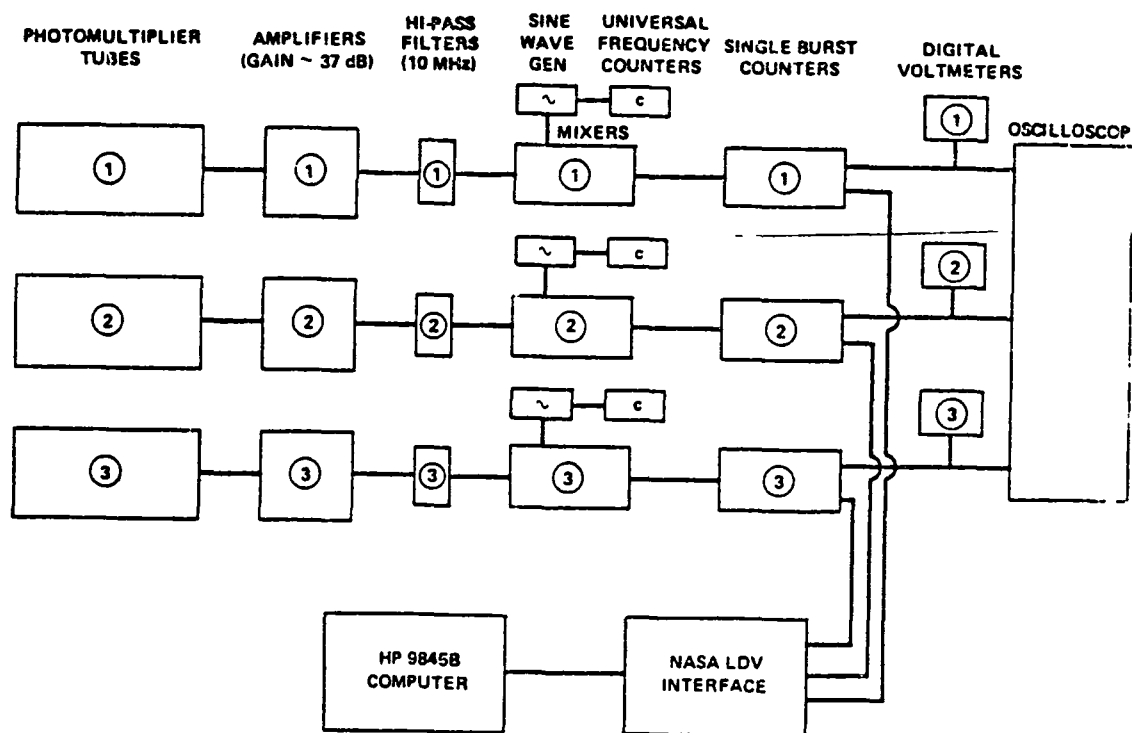


Fig. 6 Schematic of the signal processing electronics.

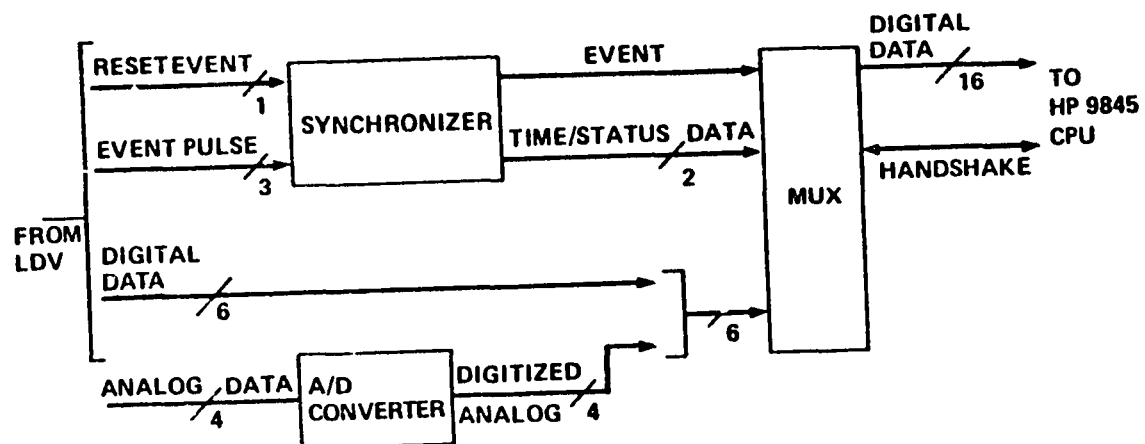


Fig. 7 Computer interface simplified block diagram.

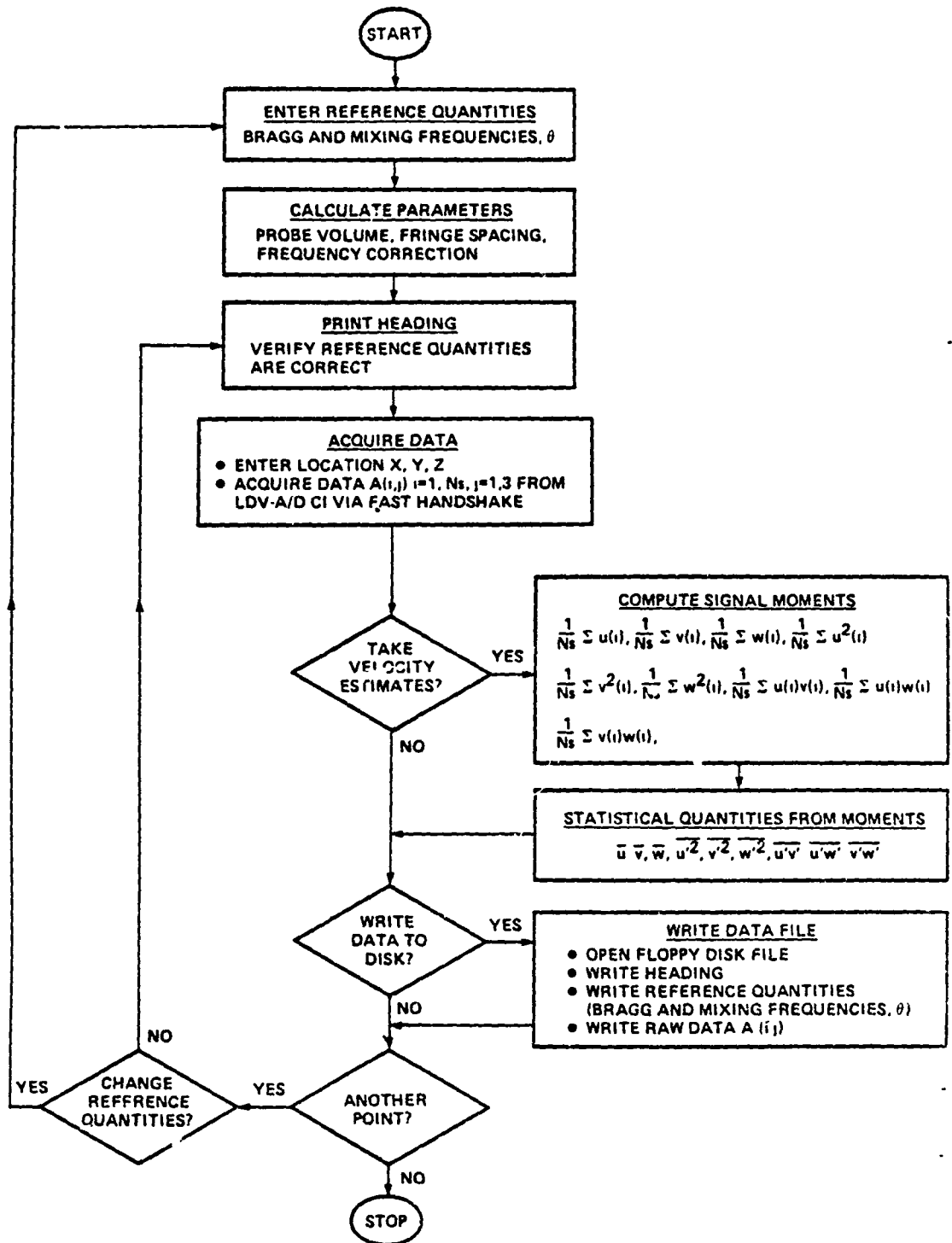


Fig. 8 Block diagram for data acquisition program.

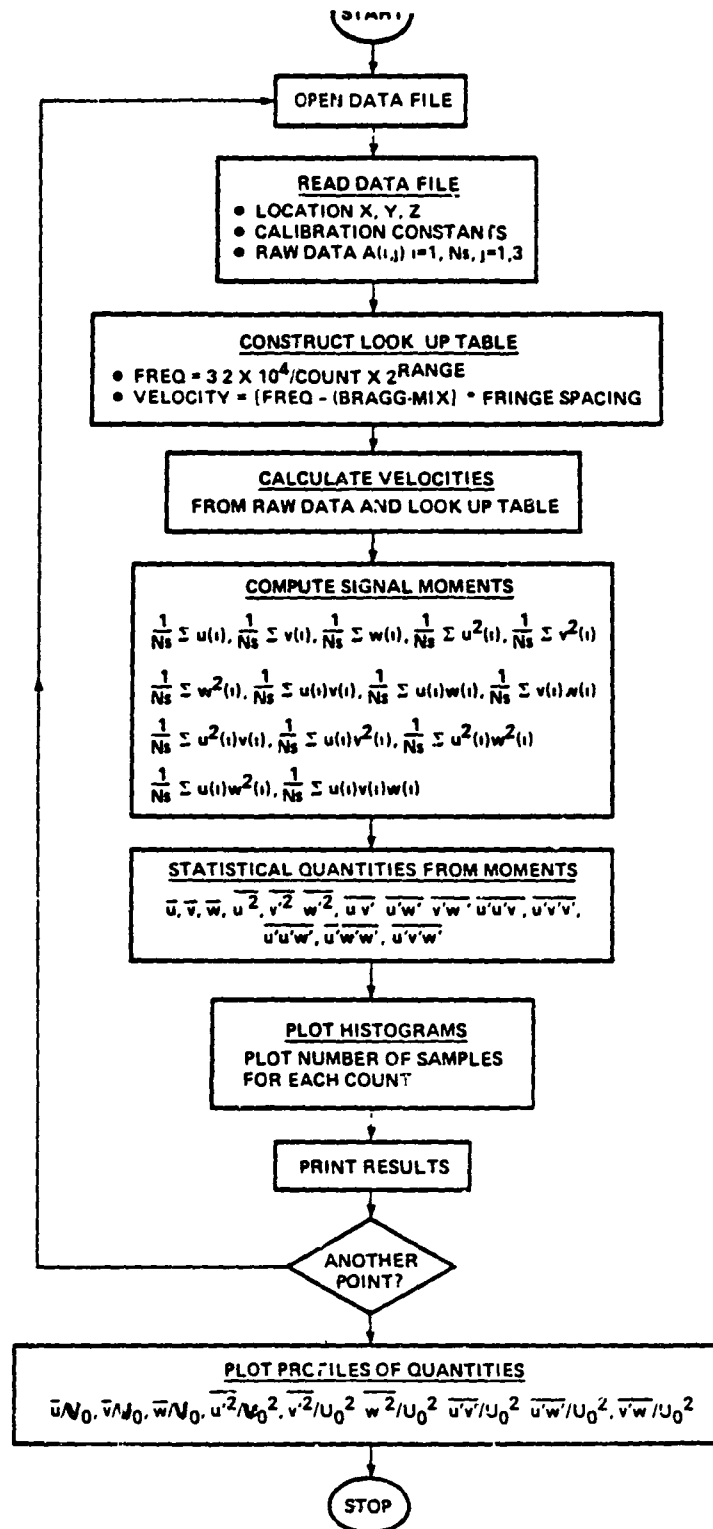


Fig. 9 Block diagram for data reduction program.

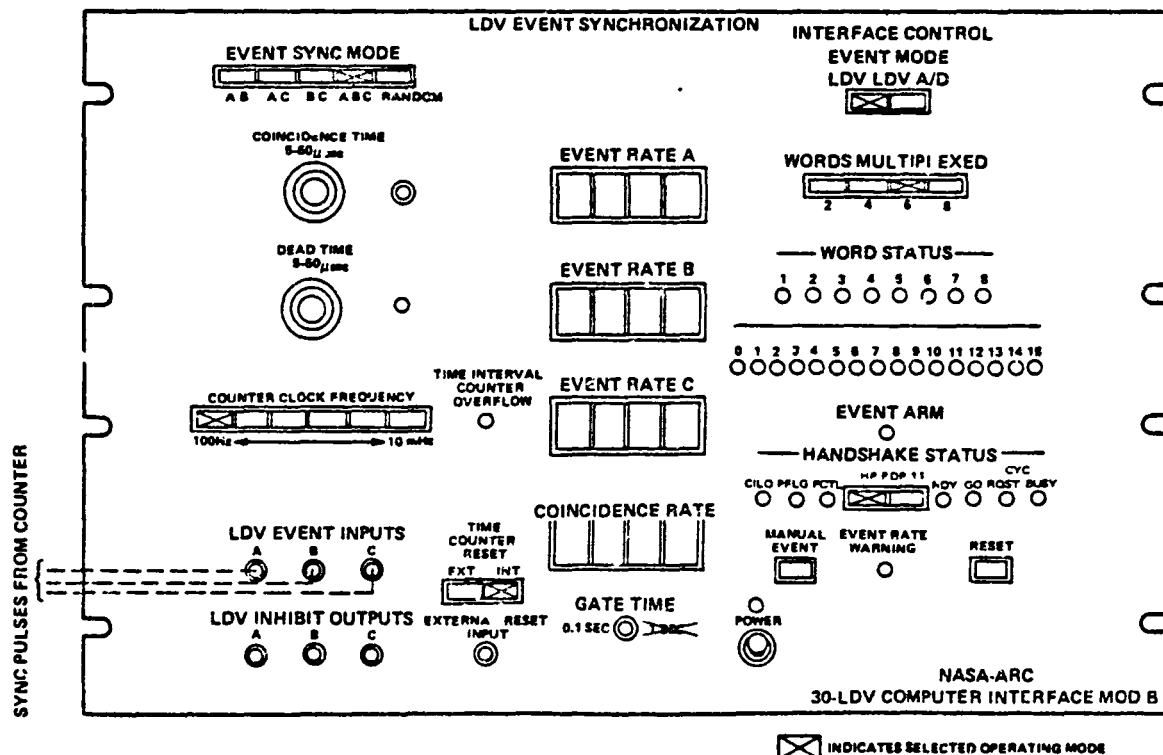
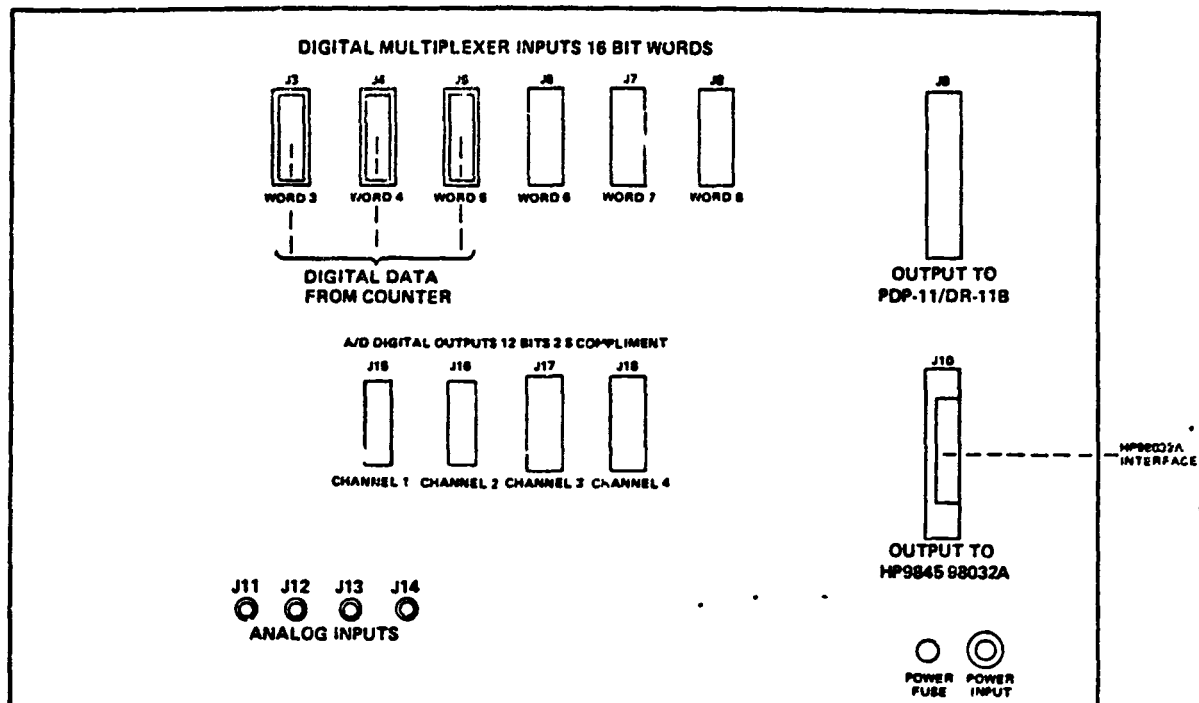
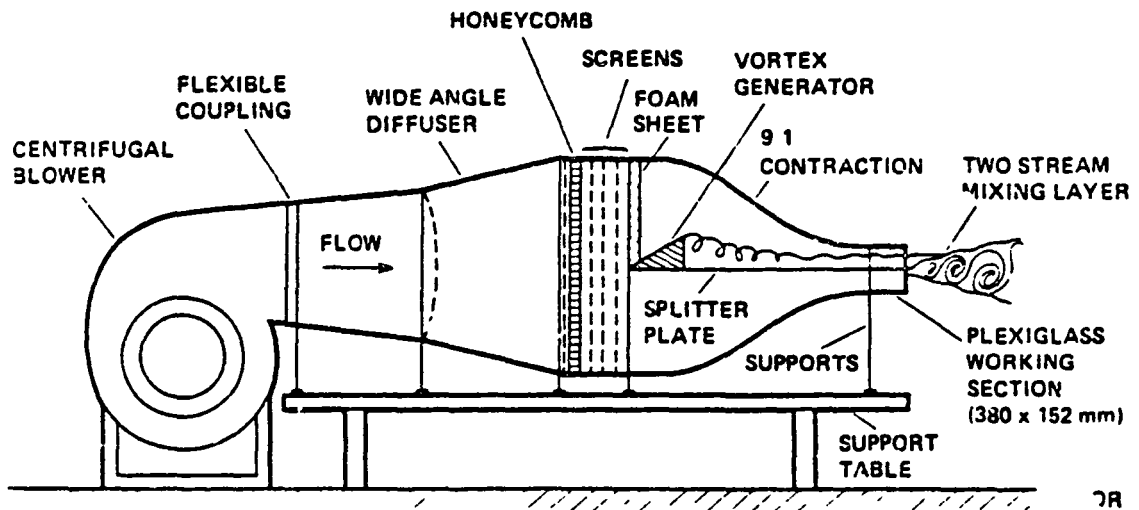
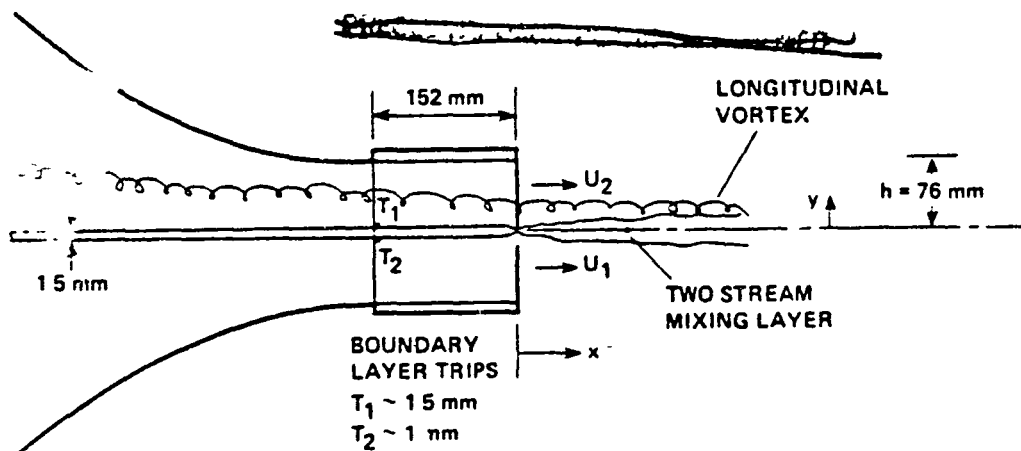


Fig. 10

NASA LDV-A/D computer interface connections and settings.

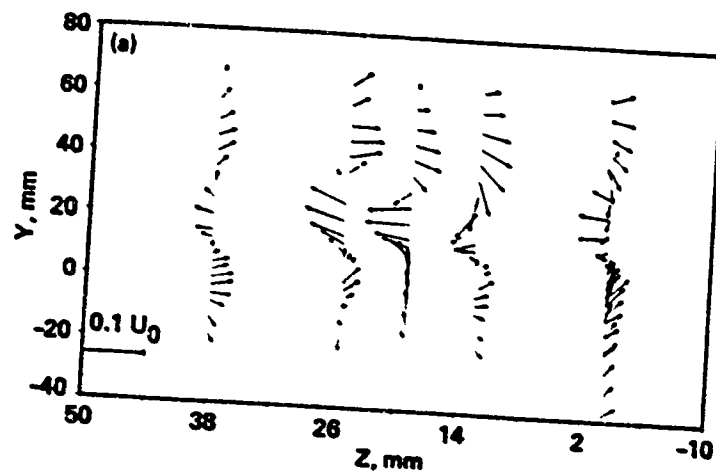


(a) Overall schematic

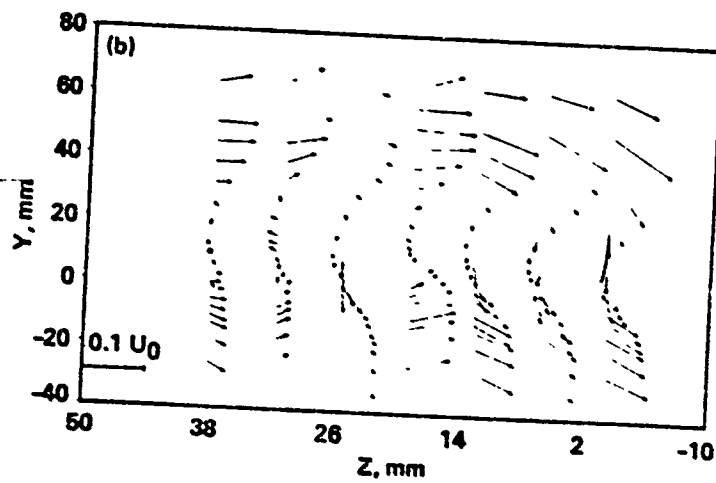


(b) Details of boundary layer trips and coordinate system

Fig. 11 Experimental rig.



(a) Cross-wire measurements



(b) LDV measurements

Fig. 12 Secondary velocity plots.

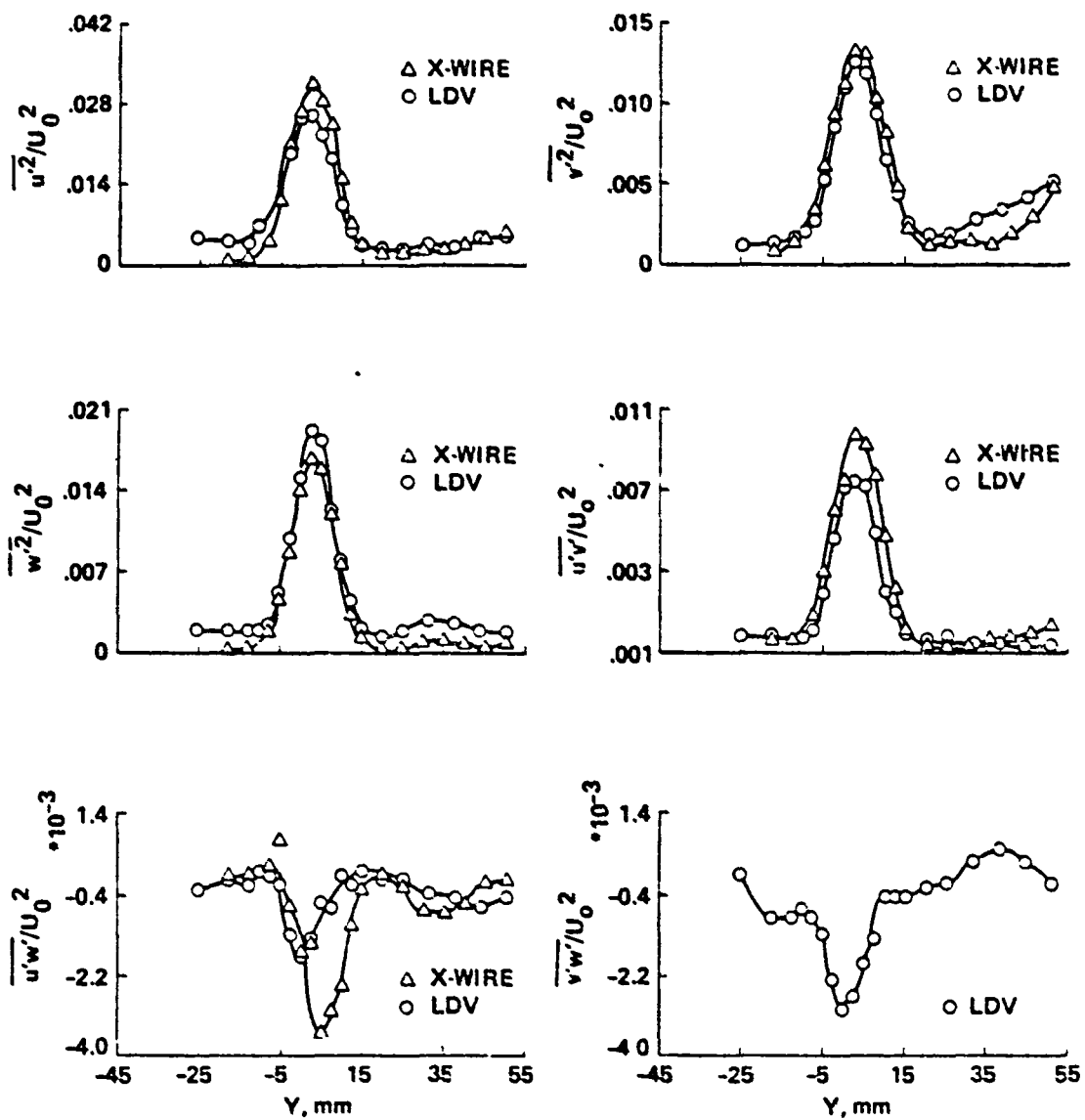


Fig. 13

Comparison of X-wire and LDV measurements in a vortex/mixing layer interaction.

**END
DATE
FILMED**

NOV 13 1986

End of Document